You have downloaded a document from
RE-BUŚ
repository of the University of Silesia in Katowice

Title: Transport in complex systems : a lattice Boltzmann approach

Author: Michał Januszewski

Citation style: Januszewski Michał. (2015). Transport in complex systems : a lattice Boltzmann approach. Praca doktorska. Katowice : Uniwersytet Śląski

© Korzystanie z tego materiału jest możliwe zgodnie z właściwymi przepisami o dozwolonym użytku lub o innych wyjątkach przewidzianych w przepisach prawa, a korzystanie w szerszym zakresie wymaga uzyskania zgody uprawnionego.

UNIWERSYTET ŚLĄSKI W KATOWICACH
Biblioteka Uniwersytetu Śląskiego
Ministerstwo Nauki i Szkolnictwa Wyższego

Uniwersytet Śląski
Wydział Matematyki Fizyki i Chemii
Instytut Fizyki

Michał Januszewski

Rozprawa doktorska

Transport in complex systems: a lattice Boltzmann approach

Promotor:
dr hab. Marcin Kostur

Katowice, 2015

Abstract

This thesis addresses the problem of effective modeling of transport processes in complex fluid dynamical systems with the lattice Boltzmann method. System complexity is approached from various perspectives, with the specific analyzed systems covering a wide range of phenomena, including multiphase flows, hemodynamics, and turbulence. In all cases, particular attention is paid to computational aspects — the accuracy of the used models, as well as the speed with which a solution can be obtained.

An open source implementation of the lattice Boltzmann method for graphics processing units called „Sailfish" is developed, benchmarked, validated, and discussed in detail. It is then used to address three flow problems.

In the first setting, the free energy multifluid model is applied to Bretherton/Taylor flows in two- and three-dimensional geometries, showing good agreement with results available in the literature, both experimental and those obtained with other numerical methods. In the second setting, the problem of blood flow in realistic geometries of human cerebral vasculature is addressed with time-dependent flow simulations. The results are carefully compared to solutions obtained with the Finite Volume Method with OpenFOAM, accelerated with a commercial GPU extension. Good agreement between the two methods is obtained, and it is shown that the lattice Boltzmann method provides a compelling option for such simulations, offering speed-ups of up to 20x. Finally, turbulent flows in simple geometries are addressed, after validation of all implemented relaxation models on the Kida vortex test case. Channel flows with and without obstacles are discussed both in fully resolved and under-resolved simulations. Good agreement with prior numerical and experimental work is again obtained, and the stability of our implementation is shown to exceed that previously reported for the same flows and relaxation models.

Streszczenie

Celem niniejszej pracy jest zbadanie możliwości efektywnego modelowania procesów transportu w złożonych systemach z zakresu dynamiki płynów za pomocą metody siatkowej Boltzmanna (LBM). Złożoność systemu została potraktowana wieloaspektowo i konkretne układy, które poddano analizie pokrywały szeroki zakres zagadnień fizycznych, m.in. przepływy wielofazowe, hemodynamikę oraz turbulencję. We wszystkich przypadkach szczególna uwaga została zwrócona na aspekty numeryczne — dokładność używanych modeli, jak również szybkość z jaką pozwalają one uzyskać zadowalające rozwiązanie.

W ramach pracy rozwinięty został pakiet oprogramowania Sailfish, będący otwartą implementacją metody siatkowej Boltzmanna na procesory kart graficznych (GPU). Po analizie szybkości jego działania, walidacji oraz omówieniu założeń projektowych, pakiet ten został użyty do symulacji trzech typów przepływów.

Pierwszym z nich były przepływy typu Brethertona/Taylora w dwu- i trójwymiarowych geometriach, do symulacji których zastosowano model energii swobodnej. Analiza otrzymanych wyników pokazała dobrą zgodność z danymi dostępnymi w literaturze, zarówno eksperymentalnymi, jak i otrzymanymi za pomocą innych metod numerycznych. Drugim badanym problemem były przepływy krwi w realistycznych geometriach tętnic dostarczających krew do ludzkiego mózgu. Wyniki symulacji zostały dokładnie porównane z rozwiązaniem otrzymanym metodą objętości skończonych z wykorzystaniem pakietu OpenFOAM, przyspieszonego komercyjną bilbioteką pozwalającą na wykonywanie obliczeń na GPU. Otrzymano dobrą zgodność między badanymi metodami oraz pokazano, że metoda siatkowa Boltzmanna pozwala na wykonywanie symulacji do ok. 20 razy szybciej. Trzecim przeanalizowanym zagadnieniem były turbulentne przepływy w prostych geometriach. Po zwalidowaniu wszystkich zaimplementowanych modeli relaksacji na przypadku wiru Kidy, zbadano przepływy w pustym kanale oraz w obecności przeszkód. Do symulacji wykorzystano zarówno siatki zapewniające pełną rozdzielczość aż do skal Kolmogorova, jak i siatki o mniejszej rozdzielczości. Również w tym kontekście pokazano dobrą zgodność wyników otrzymanych metodą siatkową Boltzmanna z wynikami innych symulacji oraz badaniami eksperymentalnymi. Pokazano również, że implementacja LBM w pakiecie Sailfish zapewnia większą stabilność obliczeń niż ta opisana w literaturze dla tych samych przepływów i modeli relaksacji.

Glossary

CFD Computational Fluid Dynamics

CPU Central Processing Unit

CVS Cardiovascular System

DNS Direct Numerical Simulation

EDM Exact Difference Method

ELBM Entropic Lattice Boltzmann Method

FE Free Energy

FEM Finite Element Method

FVM Finite Volume Method

GPGPU General-Purpose Computing on GPUs

GPU Graphics Processing Unit

HPC High Performance Computing

ICA Internal Carotid Artery

LBM Lattice Boltzmann Method

LB Lattice Boltzmann

LBGK Lattice BGK

LGA Lattice Gas Automata

MFU Minimal Flow Unit

MLUPS Millions of Lattice site Updates per Second

MRT Multiple Relaxation Times

NSE Navier-Stokes equation

OSI Oscillatory Shear Index

PDF Particle Distribution Function

RAM Random Access Memory

RBC Red Blood Cell

RLB Regularized Lattice Boltzmann

RTCG Run-Time Code Generation

SC Shan-Chen model

SIMT Single Instruction, Multiple Threads

SM Streaming Multiprocessor

SP Scalar Processor

WSS Wall Shear Stress

Contents

1 Introduction

Fluid mechanics, the branch of physics focusing on the study of fluid flows, finds itself in an interesting position. On the one hand, a wide range of fluid flows can be described using only classical mechanics. In fact, the basic governing equations — the Navier-Stokes equations (NSEs) have been known since the 19th century. On the other hand, a lot of problems remain unsolved, despite the passage of time and repeated attempts to crack them.

Despite two centuries of progress, we still do not know if three dimensional solutions of the NSEs always exist, and if they do, whether they are smooth. A $1M prize will be awarded by the Clay Mathematics Institute to one who presents a proof of this hypothesis or a counter-example. The NSEs are also believed to adequately describe turbulent flows. Yet, a full understanding of turbulence is still very much an open problem, sometimes called „the last great unsolved problem of classical physics" (a quote attributed to Richard Feynman). Another quote, variously attributed to either Horace Lamb or Weiner Heisenberg, further highlights the complexity of this challenge:

> I am an old man now, and when I die and go to heaven there are two matters on which I hope for enlightenment. One is quantum electrodynamics, and the other is the turbulent motion of fluids. And about the former I am rather optimistic.

Despite the lack of a complete theoretical understanding, fluid mechanics has enormous practical significance, with wide-ranging applications spanning such diverse fields as engineering, biology, medicine, astrophysics, and meteorology, to name just a few. While few analytical solutions of the NSEs exist, real problems are solved through the application of numerical methods in what became known as Computational Fluid Dynamics (CFD).

This thesis is focused on a particular CFD method called Lattice Boltzmann Method (LBM). Unlike more traditional approaches where the NSEs are solved directly, LBM is deeply rooted in statistical physics, addressing the problem of fluid motion at a „mesoscopic" level, a link we will briefly explore in the following chapter. In order to use the LBM to solve real problems, we took advantage of the enormous computational power presented by modern Graphics Processing Units (GPUs). While the LBM had been in use for many years before GPUs became common, this novel computing architecture makes simulations significantly easier and faster, with problems previously requiring the use of a computer cluster now being solvable with a single workstation in a shorter time.

In order to make the thesis self-contained, Chapter 2 presents an overview of the var-

ious variants of LBM relevant to this work. Relaxation models, boundary conditions, and various miscellaneous issues are discussed, together with the theoretical foundations of fluid dynamics. This is followed in Chapter 3 by a discussion of effective LBM implementation on GPUs. The chapter presents the design principles and capabilities of the *Sailfish* code, which was developed specifically for this thesis and used for all results presented therein. Its development was a crucial factor in enabling many of the larger simulations which would have been prohibitively expensive or slow with a more traditional approach using Central Processing Units (CPUs).

The remaining three chapters discuss applications of LBM and Sailfish to three classes of problems of important practical significance. Each chapter follows a specific internal structure. First, a short overview of the background information specific to the given application area is provided. This is followed by a discussion of relevant prior work, analysis of LBM simulation results, and, where applicable, a comparison to literature data.

Chapter 4 shows how the free energy lattice Boltzmann model can be applied to Bretherton/Taylor flow problems in microchannels. Such microchannels are widely used in the chemical industry for engineering of catalysed reactions. Chapter 5 discusses blood flow in realistic geometries created from CT angiographic scans of human bodies. The increasing availability of relatively cheap computational power in the form of workstations equipped with GPUs brings about the possibility of running patient-specific flow simulations directly in the doctor's office, in clinically acceptable time, for purposes of disease management, medical diagnostics and intervention planning. Finally, Chapter 6 shows the applicability of LBM models to two benchmark problems in turbulent flows — the flow in a channel, and the flow around a cuboid obstacle. The flows are addressed in both fully-resolved and under-resolved domain geometries. We are able to attain good numerical stability and correct velocity and Reynolds stress profiles at resolutions lower than previously reported. Chapter 7 provides a short summary of all obtained results and sketches some directions for future work.

2 The lattice Boltzmann method

To establish notation, as well as to put the rest of the thesis in the proper physical context, we begin with a brief reminder of some core concepts from classical hydrodynamics. This short section is followed by a more detailed discussion of various Lattice Boltzmann (LB) models used in following chapters. All of the presented models were implemented in the Sailfish software, which was developed as part of this thesis.

2.1 Fluid dynamics

The dynamics of compressible fluid flow is described by the NSE, which can be written in a general form as:

$$\rho\left(\frac{\partial\vec{u}}{\partial t}+\vec{u}\cdot\nabla\vec{u}\right)=-\nabla p+\nabla\cdot\tau+\vec{f} \tag{2.1}$$

$$\frac{\partial\rho}{\partial t}+\nabla\cdot(\rho\vec{u})=0 \tag{2.2}$$

$$\rho\left(\frac{\partial e}{\partial t}+\vec{u}\cdot\nabla e\right)=-p\nabla\cdot\vec{u}+\tau\nabla\vec{u}+\nabla\cdot(\kappa\nabla T) \tag{2.3}$$

where ρ is the density of the fluid, $\vec{u}=(u,v,w)$ is the fluid velocity, p is the pressure, τ is the anisotropic fluid stress tensor, f is an external force (e.g. gravity), e is the specific internal energy (i.e. per unit mass of fluid, not including kinetic energy), T is the temperature and $\kappa(T)$ is the thermal conductivity. It should be noted that the total fluid stress tensor includes the isotropic part related to the hydrodynamic pressure $\sigma_{ij}=-p\delta_{ij}+\tau_{ij}$.

Eq. (2.1) - Eq. (2.3) express the conservation of momentum (Newton's second law of motion), conservation of mass and conservation of energy, respectively. Eq. (2.2) is often called the *continuity equation*. The NSEs are known to accurately describe the motion of fluids at non-relativistic velocities, as long as the continuum approximation can be assumed to hold (i.e. the Knudsen number, defined as the ratio of the mean free path to a characteristic length scale, is $\ll 1$).

Eq. (2.1) - Eq. (2.3) contain more unknowns (7) than equations (5), so additional relations are necessary in order to obtain a closed system — an equation of state for pressure $p(\rho,T)$ as well as a thermodynamic equation for internal energy $e(T,p)$. The form of the tensor τ also has to be specified. Under a number of assumptions (isotropic fluid, zero

trace condition for the tensor, vanishing value for fluid at rest, Galilean invariance), the tensor can be shown to have the form [90]

$$\tau = \mu \left(\nabla \vec{u} + (\nabla \vec{u})^T \right) - \left(\frac{2}{3}\mu - \lambda \right) \mathbb{I} \nabla \cdot \vec{u} \tag{2.4}$$

where μ is the *dynamic* viscosity and λ is the *bulk*, or *volume* viscosity. In writing Eq. (2.4) we also assumed that the fluid is *Newtonian*, i.e. that the stress tensor is linearly dependent on strain.

When the fluid is *incompressible*, i.e. $\rho = \text{const}$, Eq. (2.2) simplifies to $\nabla \cdot \vec{u} = 0$, Eq. (2.3) decouples from the remaining NSE, and viscosity becomes determined only by μ.

If we introduce some characteristic velocity and length scales U_0 and L_0, the incompressible NSE can be non-dimensionalized to yield:

$$\partial_t u_i + u_j \partial_j u_i = -\partial_i p + \frac{1}{\mathrm{Re}} \partial_j \partial_j u_i \tag{2.5}$$

$$\partial_i u_i = 0 \tag{2.6}$$

where we used the pressure scale $p_0 = \rho U_0^2$ and introduced the *Reynolds number* $\mathrm{Re} = U_0 L_0 / \nu$, where $\nu = \mu / \rho$ is the kinematic viscosity. The Reynolds number is now the only parameter in the equation, and thus it can be easily seen that systems with different physical viscosities, and length, and time scales are mathematically equivalent if they are described by the same Re.

The use of such non-dimensional parameters is a common practice in the field of fluid dynamics, since it makes it possible to maintain dynamic similarity when scaling experiments and simulations. For oscillatory flows, the *Womersley number* $\alpha = L_0 \sqrt{2\pi f_0 / \nu}$ expresses the ratio of transient to viscous forces, where f_0 is the characteristic frequency. Alternatively, the *Strouhal number* $\mathrm{St} = f_0 L_0 / U_0$ can be used to describe the oscillatory character of the flow. The three numbers are connected to each other via the relation $\alpha = \sqrt{2\pi \mathrm{ReSt}}$.

2.2 Lattice Boltzmann framework overview

In the traditional approach to CFD, one begins with the NSEs in their continuous form, then discretizes the equations and solves the resulting system of algebraic equations. If this traditional approach can be called top-down, the LBM, tracing it roots to lattice gas models, takes the exact opposite, bottom-up way. In the LBM a new, so-called *mesoscopic* description of the fluid is postulated as the starting point, and only later an equivalence of the system dynamics to the macroscopic NSE-based description is shown.

Historically, this approach can be traced back to Lattice Gas Automata (LGA) — a particular variant of cellular automata describing the dynamics of solid gas particles whose

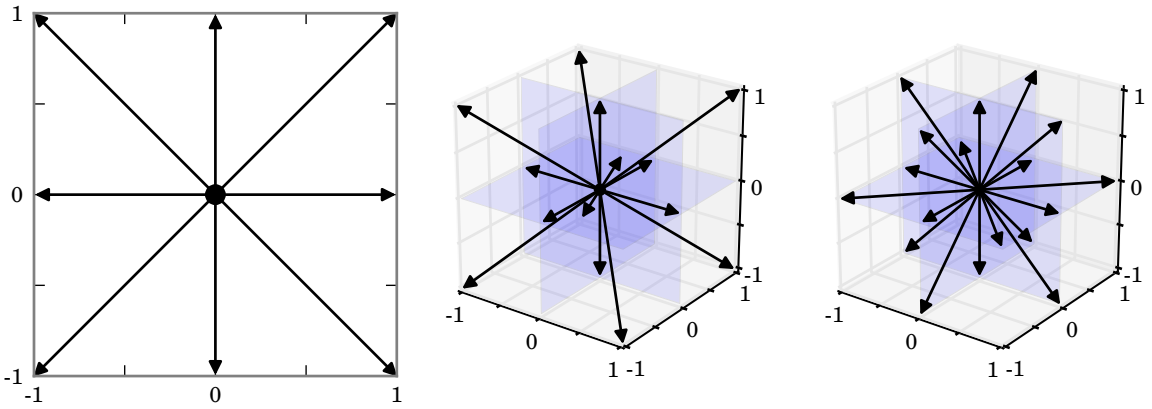


Figure 2.1: Sample lattices commonly used for LB simulations: D2Q9 (left panel), D3Q15 (middle panel), and D3Q19 (right panel). The distance unit represents the internode distance.

movement has been restricted to hopping between nodes of a spatial lattice. It was initially hoped that such models would be suitable as a description of fluid dynamics. Such a prospect appeared very attractive — since only simple rules and integer arithmetic were used in the models, it promised fast CFD simulations free from problems of numerical instabilities. After the initial enthusiasm, further research showed that the models were plagued with a number of problems, most importantly the lack of Galilean invariance and statistical noise in the results, related to the restriction to an integer number of gas particles.

The LBM was later developed as an extension of the original LGA idea. In an LB simulation, the physical space is typically discretized into a regular Cartesian grid. LB lattices are commonly referred to using the DxQy scheme, where $D = x$ indicates the dimensionality, and $Q = y$ the number of links between a node and its neighbors. Most lattices include a null link, connecting the node to itself. This link is also accounted for in Q. A few most commonly used lattices are shown in Figure 2.1. The lattice links represent discrete velocity vectors $\{\vec{e}_i, i = 0, \dots Q-1\}$, which indicate the allowed directions of mass movement between the nodes of the lattice.

Each node of the grid is associated with a small pocket of fluid. Instead of tracking the state of the fluid by the number of gas particles on the node, a Particle Distribution Function (PDF) $|f\rangle = (f_i, i = 0, \dots Q-1)$ is defined at each node. The f_i are also called *mass fractions*, since they describe the movement of mass along the corresponding discrete velocity $\vec{e}_i$. Later in the text, when we need to refer to the whole PDF at a node, we will use the Dirac bra-ket notation $|f\rangle$ in order to clearly distinguish vectors in physical space, such as $\vec{e}_i$ from vectors in the discrete velocity space of LB.

Macroscopic fluid quantities such as density (ρ), velocity ($\vec{u}$) or the momentum flux

tensor (Π) can be recovered from the moments of the PDF:

$$\rho = \sum_{i=0}^{Q-1} f_i, \qquad \rho\vec{u} = \sum_{i=0}^{Q-1} f_i\vec{e}_i, \qquad \Pi_{\alpha\beta} = \sum_{i=0}^{Q-1} f_i e_{i\alpha} e_{i\beta}. \tag{2.7}$$

In following equations the range $[0, Q-1]$ will be implied whenever a sum over the index of the discrete velocity vectors is present (unless noted otherwise, Latin indices enumerate the discrete velocities and Greek indices enumerate spatial dimensions).

2.3 Relaxation models

To complete the mesoscopic fluid description of fluid dynamics, we need to specify an equation that will control the change of f_i in time. In LBM, this evolution equation in the absence of external forces has the general form of:

$$f_i(\vec{x} + \vec{e}_i, t+1) - f_i(\vec{x}, t) = \mathcal{C}_i(|f\rangle) \tag{2.8}$$

where $\mathcal{C}$ is a collision operator, and the positions and time are expressed in the lattice unit system, in which lattice nodes are separated by one distance unit (lu) along the primary Cartesian axes, and one time unit (lt) corresponds to a single step of the simulation. At this point, we should note the similarity of Eq. (2.8) to the Boltzmann transport equation:

$$\partial_t f + v_i \partial_i f + mF_i \partial_{v_i} f = (\partial_t f)_{\text{coll}} \tag{2.9}$$

where f is a probability density function, m is the particle mass, $\vec{v}$ is the particle velocity and $\vec{F}$ is an external force. If the term related to external forces is omitted, it is rather easily seen that the Eq. (2.8) is just a discrete approximation of Eq. (2.9) [49].

Conceptually, Eq. (2.8) is often split into two parts — *collision* or *relaxation*, corresponding to the right hand side of the equation, and *propagation* or *streaming*, corresponding to the left hand side. This split provides a useful framework in which LB models can be discussed. In particular, the propagation step typically stays unchanged, while the collision can vary depending on the details of the used model.

In order to be useful in simulations, the form of the collision operator $\mathcal{C}$ needs to be explicitly specified. It is also this collision operator that will allow us to establish a connection to the macroscopic fluid dynamics of the fluid given by Eq. (2.1). In what follows, we discuss a number of progressively more complex relaxation models. The more complex models are more computationally costly, but they enhance the numerical stability of the simulation (enabling flows at higher Re without increasing lattice size), and providing additional adjustable physical parameters (e.g. bulk viscosity).

Table 2.1: Weights w_i for different lattices.

Lattice	Link length [lu]			
	0	1	$\sqrt{2}$	$\sqrt{3}$
D2Q9	4/9	1/9	1/36	-
D3Q13	1/2	-	1/24	-
D3Q15	2/9	1/9	-	1/72
D3Q19	1/3	1/18	1/36	-
D3Q27	8/27	2/27	1/54	1/216

2.3.1 LBGK

The simplest relaxation model for the LBM is the so-called Lattice BGK (LBGK) collision operator, which owes its name to the Bhatnagar-Gross-Krook operator from continuous kinetic theory [16]. With this operator, the system dynamics is described by:

$$f_i(\vec{x}+\vec{e}_i, t+1) - f_i(\vec{x}, t) = \frac{1}{\tau}\left(f_i^{\mathrm{eq}} - f_i\right)(\vec{x}, t) \tag{2.10}$$

where $|f^{\mathrm{eq}}\rangle$ is an equilibrium distribution function and the relaxation time τ is related to the kinematic viscosity ν via $\tau = (1+6\nu)/2$. The equilibrium distribution function is itself a function of the local macroscopic fluid fields ρ, $\vec{u}$ at every lattice node. When it takes the form

$$f_i^{\mathrm{eq}} = \rho w_i \left(1 + \frac{\vec{e}_i \cdot \vec{u}}{c_s^2} + \frac{(\vec{e}_i \cdot \vec{u})^2}{2c_s^4} - \frac{u^2}{2c_s^2}\right) \tag{2.11}$$

where $c_s = \sqrt{1/3}$ is the speed of sound, and when the weights w_i are taken from Table 2.1, it can be shown through Chapman-Enskog expansion [23] or expansion in a Hermite basis [127] that in the limit of low Mach numbers ($\mathrm{Ma} = u/c_s$) Eq. (2.10) reproduces the incompressible NSE dynamics with second order accuracy in the bulk of the fluid [23]. The pressure can be computed from the equation of state: $p = c_s^2\rho$. The equilibrium distribution function Eq. (2.11) yields the following moments:

$$\begin{gathered}\sum_i f_i^{\mathrm{eq}} = \rho \qquad \sum_i f_i^{\mathrm{eq}}\vec{e}_i = \rho\vec{u} \qquad \sum_i f_i^{\mathrm{eq}} e_{i\alpha}e_{i\beta} = \rho c_s^2\delta_{\alpha\beta} + \rho u_\alpha u_\beta \\ \sum_i f_i^{\mathrm{eq}} e_{i\alpha}e_{i\beta}e_{i\gamma} = \rho c_s^2\left(u_\alpha\delta_{\beta\gamma} + u_\beta\delta_{\alpha\gamma} + u_\gamma\delta_{\alpha\beta}\right)\end{gathered} \tag{2.12}$$

The Chapman-Enskog asymptotic analysis starts with a formal expansion of the distribution function and associated derivatives into a power series with a small parameter ϵ:

$$f = \sum_{n=0}^{\infty}\epsilon^n f^{(n)} \qquad \partial_t = \sum_{n=1}^{\infty}\epsilon^n\partial_{t^{(n)}} \qquad \partial_\alpha = \sum_{n=1}^{\infty}\epsilon^n\partial_{\alpha^{(n)}} \tag{2.13}$$

For the purposes of the derivation of the NSEs, the ϵ is identified with the Knudsen number, and the expansion is limited to a single order in space and two orders in time. This is motivated by the fact that macroscopic processes such as advection and diffusion take place on similar spatial scales, but can be associated with different time scales.

We first note that the left hand side of Eq. (2.10) can be Taylor expanded to yield:

$$f_i(\vec{x} + \delta\vec{e}_i, t + \delta) - f_i(\vec{x}, t) = \sum_{n=0}^{\infty} \frac{\delta^n}{n!} \left(\partial_t + \vec{e}_i \cdot \nabla\right)^n f_i(\vec{x}, t) \tag{2.14}$$

where δ is the (small) time step. A truncation of this series allows us to see the lattice Boltzmann equation as a discretization of the discrete Boltzmann equation with the LBGK collision operator:

$$\partial_t f_i + \vec{e}_i \cdot \nabla f_i = -\frac{1}{\tau}\left(f_i - f_i^{\mathrm{eq}}\right) \tag{2.15}$$

where the distribution functions are now continuous in space and time, but the velocity space is still discretized. If we now substitute suitably truncated expansions Eq. (2.13) into Eq. (2.14) truncated at second order terms, use $f_i \approx f_i^{(0)} + \epsilon f_i^{(1)} + \epsilon^2 f_i^{(2)}$, drop terms $\mathcal{O}(\epsilon^3)$ and higher and group the terms by the order in ϵ, we get:

$$f_i^{(0)} = f_i^{\mathrm{eq}} \qquad \mathcal{O}(1) \tag{2.16}$$

$$\partial_{t^{(1)}} f_i^{(0)} + \vec{e}_i \cdot \nabla^{(1)} f_i^{(0)} = -\frac{1}{\tau} f^{(1)} \qquad \mathcal{O}(\epsilon) \tag{2.17}$$

$$\partial_{t^{(2)}} f_i^{(0)} + \left(1 - \frac{1}{2\tau}\right)\left(\partial_{t^{(1)}} + \vec{e}_i \cdot \nabla^{(1)}\right) f_i^{(1)} = -\frac{1}{\tau} f^{(2)} \qquad \mathcal{O}(\epsilon^2) \tag{2.18}$$

where we used Eq. (2.17) to reach Eq. (2.18).

With Eq. (2.16) and knowing that the first two moments of f_i^{eq} yield density ρ and momentum $\rho\vec{u}$, respectively, we have to impose the constraints:

$$\sum_i f_i^{(n)} = 0, \qquad \sum_i f_i^{(n)} \vec{e}_i = 0 \qquad n \geq 1 \tag{2.19}$$

to maintain consistency with Eq. (2.7).

Taking the zeroth moment (i.e. summing over the discrete velocities) of Eq. (2.17) and Eq. (2.18) we obtain, respectively

$$\partial_{t^{(1)}} \rho + \nabla^{(1)} \rho\vec{u} = 0 \tag{2.20}$$

$$\partial_{t^{(2)}} \rho = 0 \tag{2.21}$$

which is just the continuity equation to the desired orders in space (1) and time (2).

The first moment of Eq. (2.17) and Eq. (2.18) will allow us to recover the Newtonian momentum conservation NSEs, which using Eq. (2.1) and Eq. (2.4) we can write as:

$$\partial_t \rho u_\alpha + \partial_\beta \rho u_\alpha u_\beta + \partial_\beta \rho c_s^2 \delta_{\alpha\beta} = \partial_\beta \tau_{\alpha\beta} \tag{2.22}$$

$$\tau_{\alpha\beta} = \rho\nu \left(\partial_\alpha u_\beta + \partial_\beta u_\alpha\right) - \left(\frac{2}{3}\rho\nu - \lambda\right) \partial_\gamma u_\gamma \delta_{\alpha\beta}. \tag{2.23}$$

From Eq. (2.17) we get:

$$\partial_{t^{(1)}} \rho u_\beta + \partial_{\alpha^{(1)}} \Pi^{(0)}_{\alpha\beta} = 0 \tag{2.24}$$

where $\Pi^{(0)}_{\alpha\beta} = \sum_i e_{i\alpha} e_{i\beta} f_i^{(0)}$, while from Eq. (2.18) we obtain:

$$\partial_{t^{(2)}} \rho u_\beta + \left(1 - \frac{1}{2\tau}\right) \partial_{\alpha^{(1)}} \Pi^{(1)}_{\alpha\beta} = 0 \tag{2.25}$$

where $\Pi^{(1)}_{\alpha\beta} = \sum_i e_{i\alpha} e_{i\beta} f_i^{(1)}$ and where we used Eq. (2.19). Adding Eq. (2.24) and Eq. (2.25) we see that we will be able to recover Eq. (2.22) when:

$$\Pi^{(0)}_{\alpha\beta} + \left(1 - \frac{1}{2\tau}\right) \Pi^{(1)}_{\alpha\beta} = \rho u_\alpha u_\beta + \rho c_s^2 \delta_{\alpha\beta} - \tau_{\alpha\beta}. \tag{2.26}$$

From Eq. (2.12) we can see that the first term on the left hand side of Eq. (2.26) matches the first two terms of the right hand side, leaving us with:

$$\left(\frac{1}{2\tau} - 1\right) \Pi^{(1)}_{\alpha\beta} = \rho\nu \left(\partial_\alpha u_\beta + \partial_\beta u_\alpha\right) - \left(\frac{2}{3}\rho\nu - \lambda\right) \partial_\gamma u_\gamma \delta_{\alpha\beta}. \tag{2.27}$$

We now note that using Eq. (2.17) we can write the tensor $\Pi^{(1)}$ in terms of $\Pi^{(0)}$:

$$\begin{aligned}
\Pi^{(1)}_{\alpha\beta} &= \sum_i e_{i\alpha} e_{i\beta} f_i^{(1)} \\
&= -\tau \sum_i e_{i\alpha} e_{i\beta} \left(\partial_{t^{(1)}} + e_{i\gamma}\partial_\gamma\right) f_i^{(0)} \\
&= -\tau \partial_{t^{(1)}} \Pi^{(0)}_{\alpha\beta} - \tau c_s^2 \partial_\gamma \rho \left(u_\alpha \delta_{\beta\gamma} + u_\beta \delta_{\alpha\gamma} + u_\gamma \delta_{\alpha\beta}\right) \\
&= -\tau \partial_{t^{(1)}} \rho u_\alpha u_\beta - \tau c_s^2 \delta_{\alpha\beta} \left(\partial_{t^{(1)}} \rho + \partial_\gamma \rho u_\gamma\right) - \tau c_s^2 \left(\partial_\beta \rho u_\alpha + \partial_\alpha \rho u_\beta\right) \\
&= -\tau \left(u_\alpha u_\beta \partial_{t^{(1)}} \rho + \rho \partial_{t^{(1)}} u_\alpha u_\beta + c_s^2 \rho \left(\partial_\alpha u_\beta + \partial_\beta u_\alpha\right) + c_s^2 u_\alpha \partial_\beta \rho + c_s^2 u_\beta \partial_\alpha \rho\right) \\
&= -\tau \left(c_s^2 \rho \left(\partial_\alpha u_\beta + \partial_\beta u_\alpha\right) + u_\alpha \left(c_s^2 \partial_\beta \rho + \rho \partial_{t^{(1)}} u_\beta\right) + u_\beta \left(c_s^2 \partial_\alpha \rho + \rho \partial_{t^{(1)}} u_\alpha\right)\right)
\end{aligned} \tag{2.28}$$

where we used Eq. (2.20) and dropped terms $\mathcal{O}(\mathrm{Ma}^3)$. Using Eq. (2.24) we can now substitute $c_s^2 \partial_\beta \rho$, getting:

$$\begin{aligned}
c_s^2 \partial_\beta \rho + \rho \partial_{t^{(1)}} u_\beta &= \rho \partial_{t^{(1)}} u_\beta - \rho \partial_{t^{(1)}} u_\beta - u_\beta \partial_{t^{(1)}} \rho - \partial_\alpha \rho u_\alpha u_\beta \\
&= u_\beta \partial_\alpha \rho u_\alpha - u_\beta \partial_\alpha \rho u_\alpha - \rho u_\alpha \partial_\alpha u_\beta \\
&= -\rho u_\alpha \partial_\alpha u_\beta
\end{aligned} \tag{2.29}$$

In Eq. (2.28), the subexpression Eq. (2.29) is multiplied by another u factor, making the whole expression $\mathcal{O}(\mathrm{Ma}^3)$. Dropping it, and assuming $\lambda = 2\mu/3$, we are left with

$$\left(\tau - \frac{1}{2}\right) c_s^2 = \nu \tag{2.30}$$

which links the LB relaxation time with the fluid viscosity and completes the analysis, showing that the LBE reproduces the NSEs in the macroscopic limit under the assumption that the Mach number of the flow is small.

2.3.2 Incompressible LBGK

Even though the model presented in the previous section is in practice used to simulate incompressible flows (the incompressible limit is equivalent to the limit of low Mach number), the LBGK dynamics introduces a compressibility error. Alternative models exist, which attempt to decrease this error. One possible approach is to split the density into two terms $\rho = \rho_0 + \delta\rho$ — an average value ρ_0 and a fluctuation $\delta\rho$. By substituting this into Eq. (2.11) and neglecting terms of the order of $\mathcal{O}(\mathrm{Ma}^3)$ or higher, a new equilibrium can be derived [50]:

$$f_i^{\mathrm{eq}} = w_i \left[\rho + \rho_0 \left(\frac{\vec{e}_i \cdot \vec{u}}{c_s^2} + \frac{(\vec{e}_i \cdot \vec{u})^2}{2c_s^4} - \frac{u^2}{2c_s^2}\right)\right]. \tag{2.31}$$

This new model also slightly changes the way the velocity is calculated to be:

$$\rho_0 \vec{u} = \sum_i f_i \vec{e}_i. \tag{2.32}$$

It can be shown that this model satisfies Eqs. (2.5) and (2.6) for $\mathrm{Ma} \ll 1$ and $Tc_s \gg L$, where T is the time during which the flow undergoes a macroscopic change and L is the distance on which this change takes place.

2.3.3 Multiple Relaxation Times

The LBGK operator on the right hand side of Eq. (2.10) can be interpreted as a diagonal matrix $S = \omega I$ with $\omega = 1/\tau$ acting on the non-equilibrium part of the distribution $-|f^{\mathrm{neq}}(\vec{x}, t)\rangle = |f^{\mathrm{eq}}(\vec{x}, t)\rangle - |f(\vec{x}, t)\rangle$. The core observation underlying the Multiple Relaxation Times (MRT) model is that the collision operator can in principle act in any basis. The representation of the distribution function in a different basis can be found using an invertible matrix M: $|m\rangle = M\,|f\rangle$.

It is particularly interesting to choose the basis such that $|m\rangle$ becomes a vector of moments of the particle distribution $|f\rangle$. We can then use a different diagonal matrix

$\hat{S}$ as the collision operator in the moment basis, and assign different relaxation times τ_i to different moments. In particular, elements of the collision operator $s_i = 1/\tau_i$ corresponding to conserved quantities such as density or momentum, will have to be set to 0. Some of the remaining parameters are related to the kinematic and bulk viscosities of the fluid. Other ones can be tuned to adjust the numerical stability of the simulation without affecting its physical results. Once the collision is performed, the representation of the distribution function has to be changed back to the original space in order to allow the streaming step to be performed. All in all, the MRT collision operator can be expressed as [32]:

$$M^{-1}\hat{S}\left(|m^{\mathrm{eq}}(\vec{x},t)\rangle - |m(\vec{x},t)\rangle\right) \tag{2.33}$$

The form of the matrix M is different for every lattice, and we refer the reader to the original paper of D'Humieres et al. [32] for the details. That paper also contains a detailed discussion of choices of the adjustable collision matrix elements that maximize the numerical stability of the model. A particularly interesting case is presented by the D3Q13 lattice, for which an LBGK model does not exist, but an MRT one does [31].

2.3.4 Regularized

The regularized relaxation model [91] (also known as Regularized Lattice Boltzmann (RLB)) is based on the observation that the momentum and continuity equations in the hydrodynamic regime are fully described by the hydrodynamic variables Eq. (2.7), which are less numerous (6 in 2D and 10 in 3D) than the number of degrees of freedom provided by the distribution functions. This makes it possible to replace the set of f_i with another one that leads to the same hydrodynamic variables as the original one — an idea already exploited in the previously discussed MRT model.

RLB proposes to replace the non-equilibrium part of the distributions $f_i^{\mathrm{neq}} = f_i - f_i^{\mathrm{eq}}$ with a new first order regularized value f_i^{reg} derived from the non-equilibrium stress tensor

$$\Pi_{\alpha\beta}^{\mathrm{neq}} = \sum_i f_i^{\mathrm{neq}} e_{i\alpha} e_{i\beta} \tag{2.34}$$

as

$$f^{\mathrm{reg}} = \frac{w_i}{2c_s^4}(e_{i\alpha}e_{i\beta} - c_s^2\delta_{\alpha\beta})\Pi_{\alpha\beta}^{\mathrm{neq}}. \tag{2.35}$$

The construction of the regularized distribution guarantees that the distribution function maintains the required symmetries — the corresponding hydrodynamic variables are unchanged. The regularization step is applied prior to the collision part of the LB algorithm, which then proceeds normally.

2.3.5 Entropic

The Entropic Lattice Boltzmann Method (ELBM) [68, 69] attempts to increase the stability of the LB simulation through the use of a discrete H-function $H(f) = \sum_i f_i \ln(f_i/w_i)$ and an equilibrium distribution function defined as the extremum of $H(f)$ under constraints of mass and momentum conservation:

$$f_i^{\mathrm{eq}} = \rho w_i \prod_{\alpha=1}^{D} \left(2 - \sqrt{1+u_\alpha^2}\right) \left(\frac{2u_\alpha + \sqrt{1+3u_\alpha^2}}{1-u_\alpha}\right)^{e_{i\alpha}} \tag{2.36}$$

where D is the dimensionality of the lattice.

In the relaxation step, a new dynamically adjusted parameter α is introduced:

$$f_i(\vec{x}+\vec{e}_i, t+1) - f_i(\vec{x}, t) = \omega_0 \frac{\alpha}{2} \left(f_i - f_i^{\mathrm{eq}}\right)(\vec{x}, t) \tag{2.37}$$

with $\omega_0 = 1/\tau$. α is evaluated at every simulation step as the non-trivial solution of the H-function constancy constraint:

$$H(f) = H\left(f - \alpha(f - f^{\mathrm{eq}})\right). \tag{2.38}$$

The argument of the H function on the right hand side of Eq. (2.38) is the maximally changed post-relaxation state, retaining the same entropy as that of the pre-collision state. It should be noted that when $\alpha \approx 2$, which is the case when the system is close to equilibrium, Eq. (2.37) has the same form as Eq. (2.10). The corrections resulting from H-theorem compliance can be both positive and negative, temporarily increasing and decreasing the effective local viscosity of the fluid [2, 26].

The corrections are typically applied to a small fraction of nodes at every step of the simulation. This makes it possible to employ a number of optimization techniques aiming at reducing the computational cost of ELBM. Some of those proposed in the literature include using an asymptotic expansion for α when the distribution is close to the equilibrium [26], assuming $\alpha = 2$ for nodes close to equilibrium, or using the α estimate from the previous time step as the starting point for a numerical search of the root of Eq. (2.38) [142].

ELBM can in principle guarantee unconditional stability of the numerical scheme, as the $H(f)$ function can be proven to be a Lyapunov function for the system. At sufficiently low lattice sizes or high Reynolds numbers the simulation will however still generate unphysical results, so additional care has to be taken to keep it in a regime where this is not an issue.

2.4 Multi-fluid models

A number of approaches for introducing multiple fluid species within the LB framework exist. The most commonly used ones include the red-blue model [43], the Shan-Chen model and the free-energy model. The last two are implemented in Sailfish and are diffuse interface models, where no explicit tracking of the interface is necessary, as the location of various fluid species emerges naturally from the underlying model dynamics. Both models are briefly discussed below.

2.4.1 Shan-Chen

In the Shan-Chen model [126], every fluid species is associated with its own particle distribution function, and both functions are defined on the same lattice. In what follows, we will restrict the discussion to a two-component model, where f_i describes component A, and g_i describes component B.

The equilibrium function, relaxation scheme, and density calculation follow the LBGK model. The velocity of the fluid becomes a weighted average of the first moments of the distribution functions:

$$\vec{u} = \frac{\rho_A \vec{u}_A/\tau_A + \rho_B \vec{u}_B/\tau_B}{\rho_A/\tau_A + \rho_B/\tau_B},$$

where $\vec{u}_A$, $\vec{u}_B$ and ρ_A, ρ_B are velocities and densities computed respectively from f_i and g_i using Eq. (2.7), and τ_A, τ_B are relaxation times for the two fluid components.

The coupling between the two fluid components is realized by introducing a body force into Eq. (2.10) for the first component:

$$\vec{F}_A(\vec{x}) = G\psi_A(\vec{x}) \sum_i w_i \psi_B(\vec{x} + \vec{e}_i)\vec{e}_i, \tag{2.39}$$

where $\psi_B(\vec{x})$ is a *pseudopotential* function of density ρ_B at node $\vec{x}$, and G is a coupling constant. A similar term $\vec{F}_B$ with the indices A replaced by B and vice versa, is added to the collision operator of the second fluid component. The pseudopotential function is commonly chosen as $\psi(\rho) = 1 - e^{-\rho}$.

2.4.2 Free energy

The free energy model is based on a Landau free energy functional for a binary fluid [118, 134]. Similarly to the Shan-Chen model, two PDFs are used, but g_i represents an order parameter ϕ smoothly varying between -1 (pure second component) and 1 (pure first component). Macroscopic fluid variables are thus defined as

$$\rho = \sum_i f_i \qquad \phi = \sum_i g_i \qquad \rho\vec{u} = \sum_i \vec{e}_i f_i. \tag{2.40}$$

The macroscopic dynamics of the fluid is described by

$$\begin{aligned} \partial_t \rho + \partial_\alpha(\rho u_\alpha) &= 0 \\ \partial_t(\rho u_\beta) + \partial_\alpha(\rho u_\alpha u_\beta) &= -\partial_\alpha P_{\alpha\beta} + \partial_\alpha\{\nu\rho(\partial_\beta u_\alpha + \partial_\alpha u_\beta)\} \\ \partial_t \phi + \partial_\alpha(\phi u_\alpha) &= M\nabla^2\mu, \end{aligned} \tag{2.41}$$

where M is a mobility parameter, μ is the chemical potential and the pressure tensor $P_{\alpha\beta}$ is defined as:

$$P_{\alpha\beta} = \left(p_0 - \kappa\phi\nabla^2\phi - \frac{\kappa}{2}|\nabla\phi|^2\right)\delta_{\alpha\beta} + \kappa\partial_\alpha\phi\partial_\beta\phi, \tag{2.42}$$

with $p_0 = c_s^2\rho + a\left(-\phi^2/2 + 3\phi^4/4\right)$ being the bulk pressure.

The LBM implementation of this model uses LBGK relaxation on both lattices with custom equilibrium functions designed to recover Eq. (2.41) in the macroscopic limit [118]:

$$f_i^{\mathrm{eq}} = w_i\left(p_0 - \kappa\phi\nabla^2\phi + e_{i\alpha}u_\alpha\rho + \frac{1}{2c_s^2}\left[e_{i\alpha}e_{i\beta} - c_s^2\delta_{\alpha\beta}\right]\rho u_\alpha u_\beta\right) + \kappa w_i^{\alpha\beta}\partial_\alpha\phi\partial_\beta\rho \tag{2.43}$$

$$g_i^{\mathrm{eq}} = w_i\left(\Gamma\mu + e_{i\alpha}u_\alpha\phi + \frac{1}{2c_s^2}\left[e_{i\alpha}e_{i\beta} - c_s^2\delta_{\alpha\beta}\right]\phi u_\alpha u_\beta\right) \tag{2.44}$$

for $i = 1, \ldots, Q-1$ and $f_0^{\mathrm{eq}} = \rho - \sum_{i=1}^{Q-1} f_i^{\mathrm{eq}}$, $g_0^{\mathrm{eq}} = \phi - \sum_{i=1}^{Q-1} g_i^{\mathrm{eq}}$.

Here Γ is a tunable parameter related to mobility via $M = \Gamma(\tau_\phi - 1/2)$, and the chemical potential is defined as $\mu = a(-\phi + \phi^3) - \kappa\nabla^2\phi$. κ and a are constant parameters related to surface tension $\gamma = \sqrt{8\kappa a/9}$ and interface width $\xi \propto \sqrt{\kappa/a}$. The values of the weights w_i and $w_i^{\alpha\beta}$ can be found in the paper by Kusumaatmaja and Yeomans [86].

The two fluid components can have different viscosities and corresponding relaxation times τ_A and τ_B. For f_i collisions, an effective linearly interpolated relaxation time $\tau_\rho = \tau_B + \frac{\phi+1}{2}(\tau_A - \tau_B)$ is used. The order parameter PDF g_i uses an independent relaxation time τ_ϕ which is a tunable parameter.

The coupling between f_i and g_i is provided indirectly through the fluid velocity $\vec{u}$, as well as through the presence of ϕ and its first and second-order spatial derivatives in Eq. (2.43). These derivatives are evaluated using standard finite difference schemes. In order to minimize spurious currents at the interface between the two fluid components, optimized gradient and Laplacian stencils can be used [117].

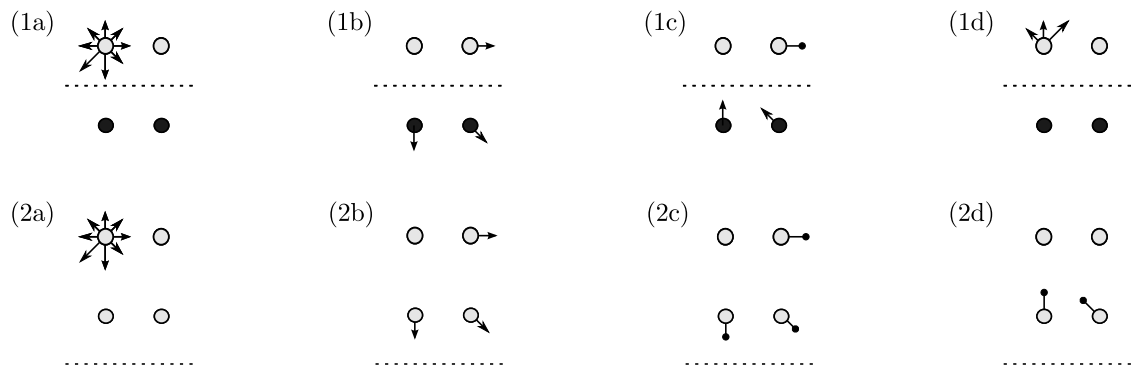


Figure 2.2: Two bounce-back schemes illustrated for the D2Q9 lattice: (1) half-way bounce-back, (2) bounce-back on the link. The top row represents fluid nodes (for which macroscopic quantities are defined and both relaxation and streaming are performed). Nodes in the bottom row are bounce-back nodes. Black nodes are „dry" (do not represent fluid, do not undergo relaxation), while grey nodes are „wet". The dashed line represents the effective location of the no-slip boundary condition. For simplicity, only distributions originating from the top left node are shown in subsequent time steps. The steps are: (a) initial state, time t; (b) streaming, time $t + 1$; (c) relaxation, time $t + 1$ (arrows ending with disks represent the post-relaxation state); (d) streaming, time $t + 2$.

2.5 Boundary conditions

2.5.1 No-slip

The no-slip boundary condition states that the fluid should have no velocity relative to a solid boundary along which it is flowing. In LB simulations, it is typically implemented through a variant of the *bounce-back* rule. This rule is conceptually very simple — incoming distributions are reflected back in the direction from which they came. The simplicity of implementation of this and other simple boundary conditions is often quoted as a large advantage of the LBM as a method for simulating fluid flows.

A number of variations of the original idea are present in the literature, differing in implementation details, conceptual complexity, and numerical stability. In what follows, we discuss some of the most popular variants used in later chapters.

Half-way bounce-back

The simplest implementation of the bounce-back condition is the *complete* or *full* bounce-back (FBB) [161]

$$f_i(x,t) = f_{-i}(x,t) \tag{2.45}$$

where all distributions are replaced, no relaxation is performed on the nodes and $-i$ represents the direction opposite to i. If the wall is interpreted to be placed half-way between the bounce-back row and the first fluid row, this scheme can be proved to be of second order accuracy [51]. Somewhat confusingly, even though the underlying procedure does not change at all, it is then called *half-way* bounce-back (HBB).

Bounce-back on the link

Ladd [88] introduced the bounce-back *on the link* (BBL) condition defined as:

$$f_i(x,t+1) = f_{-i}(x,t_+) \tag{2.46}$$

where (r,t_+) is the local *postcollision* state. Here, the bounce-back nodes undergo standard collisions and only unknown distributions are replaced using Eq. (2.46). Unknown distributions are those that point into the domain at the boundary node. The BBL scheme is also of second order accuracy if the wall is assumed to be half a distance unit beyond the bounce-back node.

Tamm-Mott-Smith

In the Tamm-Moth-Smith (TMS) boundary condition [24], the missing distributions $i \in U$ are first updated using the FBB rule Eq. (2.45), and used to compute the *target* macroscopic variables for the node:

$$\rho_{\mathrm{tg}} = \sum_{i\in K} f_i + \sum_{i\in U} f_{-i} \qquad \rho_{\mathrm{tg}}\vec{u}_{\mathrm{tg}} = \sum_{i\in K} \vec{e}_i f_i + \sum_{i\in U} \vec{e}_i f_{-i}. \tag{2.47}$$

where K denotes the set of indices of known distributions. These target values can then be used to compute the instantaneous density ρ and velocity $\vec{u}$:

$$\rho = \sum_{i\in K} f_i + \sum_{i\in U} f_i^{\mathrm{eq}}(\rho_{\mathrm{tg}}, \vec{u}_{\mathrm{tg}}) \qquad \rho\vec{u} = \sum_{i\in K} \vec{e}_i f_i + \sum_{i\in U} \vec{e}_i f_i^{\mathrm{eq}}(\rho_{\mathrm{tg}}, \vec{u}_{\mathrm{tg}}). \tag{2.48}$$

The populations at the boundary node are finally overridden as $f_i^\star = f_i^{\mathrm{eq}}(\rho_{\mathrm{tg}}, \vec{u}_{\mathrm{tg}}) + f_i^{\mathrm{neq}}$, with the non-equilibrium part defined as:

$$f_i^{\mathrm{neq}} = f_i^{\mathrm{eq}}(\rho_{\mathrm{tg}}, \vec{u}_{\mathrm{tg}}) - f_i^{\mathrm{eq}}(\rho, \vec{u}) \qquad \text{for } i \in U \tag{2.49}$$

$$f_i^{\mathrm{neq}} = f_i - f_i^{\mathrm{eq}}(\rho, \vec{u}) \qquad \text{for } i \in K. \tag{2.50}$$

and undergo standard relaxation and propagation of Eq. (2.10). When the definition Eq. (2.50) is substituted into Eq. (2.10), it can be seen that the missing distributions become a linear combination of two equilibria — one at $(\rho, \vec{u})$, and one at $(\rho_{\text{tg}}, \vec{u}_{\text{tg}})$, which provides the motivation for the name of this boundary condition due to its similarity to the Tamm-Mott-Smith approximation for the shock wave solution of the Boltzmann equation.

Alternatively, the TMS procedure can be viewed as replacing the missing populations with $f_i^{\text{eq}}(\rho_{\text{tg}}, \vec{u}_{\text{tg}})$, and then proceeding with standard LBGK relaxation of the whole PDF towards the instantaneous equilibrium $f_i^{\text{eq}}(\rho, \vec{u})$, with an additional forcing term $f_i^{\text{eq}}(\rho_{\text{tg}}, \vec{u}_{\text{tg}}) - f_i^{\text{eq}}(\rho, \vec{u})$.

2.5.2 Velocity and pressure

Velocity and pressure boundary conditions are implemented similarly for the case of an axis-aligned boundary plane. Below, we provide a detailed discussion of the equilibrium and regularized boundary conditions, used elsewhere in the text.

First, a discrete boundary normal vector (pointing outside of the fluid domain)

$$\vec{n} = \{(0, 0, \pm 1), (0, \pm 1, 0), (\pm 1, 0, 0)\}$$

is externally provided when setting up the boundary condition or detected automatically by a local analysis of the geometry. For every boundary node, a subset of distributions $U = \{f_i, \vec{e}_i \cdot \vec{n} = -1\}$ is going to be undefined. Visually, these are the distributions that point *into* the simulation domain. We note that the macroscopic variables at the boundary node can be defined as:

$$\rho = f_{n+} + f_{n-} + f_{n\perp} \qquad \rho u_n = f_{n+} - f_{n-} \tag{2.51}$$

where we grouped the distributions into sums of over discrete velocities with a component parallel to the normal vector (f_{n+}), antiparallel (f_{n-}), and perpendicular to it ($f_\perp$). One of f_{n+}, f_{n-} is going to be composed entirely of the undefined distributions U, and with one of ρ or u_n provided externally, the other can be computed by solving the system Eq. (2.51) (the contribution from the unknown distributions can be canceled out).

After the initial stage of macroscopic variable determination, which is common to both boundary condition implementations, the treatment becomes different. For the equilibrium boundary condition, an equilibrium distribution is computed using the macroscopic variables and assigned to the boundary node.

For the regularized boundary condition, the unknown distributions are defined through the „bounce-back of non-equilibrium parts", i.e.

$$f_i = f_{-i} - f_{-i}^{\text{eq}} + f_i^{\text{eq}}, \qquad i \in U \tag{2.52}$$

and then all distributions on the boundary node are rewritten using the regularized distribution Eq. (2.35) $f_i = f_i^{\text{reg}}$.

2.6 Miscellaneous

2.6.1 Body forces

A large number of schemes for adding an external force $\vec{F}$ to Eq. (2.8) have been proposed in the literature. Many of them, while conceptually simple, suffer from the presence of spurious terms (when compared to the NSE) in the macroscopic equations that can be derived from them through the Chapman-Enskog expansion. Here we discuss two widely used schemes that minimize these terms: Guo's method [44], and the Exact Difference Method (EDM) [84].

Both schemes modify the way the macroscopic fluid velocity is computed. In order to avoid using the same variable name for two different quantities the new velocity is denoted $\vec{v}$ and can be computed as

$$\rho\vec{v} = \sum_i \vec{e}_i f_i + \vec{F}/2 = \rho\vec{u} + \vec{F}/2. \tag{2.53}$$

Both schemes also propose to add a new force term F_i to the right hand side of Eq. (2.8). Guo et al. [44] analyzed a number of popular schemes and concluded [44] that in order to match the NSE, the force term should take the form:

$$F_i = \left(1 - \frac{1}{2\tau}\right) w_i \left[\frac{\vec{e}_i - \vec{v}}{c_s^2} + \frac{\vec{e}_i \cdot \vec{v}}{c_s^4}\vec{e}_i\right] \cdot \vec{F}. \tag{2.54}$$

Kupershtokh et al. [84] however showed in a simulation of the coexistence of vapor and liquid with a fluid with the van der Waals equation of state that Eq. (2.54) leads to a vapor density result noticeably different from the theoretical prediction. Their proposed solution is an alternative implementation of forces known as the EDM. In EDM, the body force term takes the form of a difference of two equilibrium distribution functions, computed using momentum after and before the action of the force $\vec{F}$:

$$F_i = f_i^{\text{eq}}(\rho, \vec{u} + \Delta\vec{u}) - f_i^{\text{eq}}(\rho, \vec{u}) \tag{2.55}$$

with $\Delta\vec{u} = \vec{F}/\rho$. The advantages of the EDM are its lack of any spurious terms in the macroscopic NSE and its applicability to any collision operator (not only LBGK). The main disadvantage is computational complexity.

2.6.2 Round-off minimization

The LBGK equilibrium function has the form $f_i^{\mathrm{eq}} = w_i\rho(1 + F(\vec{u}))$, which contains the sum of two terms of different orders of magnitude — $\mathcal{O}(1)$ and $\mathcal{O}(\mathrm{Ma})$. The same observation holds for the distribution function in general. This leads to unavoidable precision loss when the value is represented using floating point numbers. The idea of the round-off minimization model, originally presented by Skordos [128] and discussed in more detail by Dellar [30], is to change the distribution so that it only contains terms $\mathcal{O}(\mathrm{Ma})$:

$$h_i = f_i - \rho_0 w_i. \tag{2.56}$$

ρ_0 comes from the density decomposition discussed previously in the context of the incompressible model (see Section 2.3.2): $\rho = \rho_0 + \delta\rho$. Here, the average value ρ_0 is typically chosen close to 1, and the deviation term $\delta\rho$ is $\mathcal{O}(\mathrm{Ma}^2)$. With the new distributions defined in Eq. (2.56), the calculation of macroscopic variables has to be adjusted accordingly:

$$\delta\rho = \sum_i h_i, \qquad \rho = \rho_0 + \sum_i h_i, \qquad \vec{u} = \frac{\sum_i h_i \vec{e}_i}{\rho}, \qquad \Pi_{\alpha\beta} = \rho_0 c_s^2 \delta_{\alpha\beta} + \sum_i h_i e_{i\alpha} e_{i\beta}. \tag{2.57}$$

The validity of Eq. (2.57) can be verified simply by substitution of Eq. (2.56) into Eq. (2.7). To complete the model, the equilibrium function is also redefined as $h_i^{\mathrm{eq}} = f_i^{\mathrm{eq}} - w_i\rho_0$, while the relaxation operator retains its original form Eq. (2.10), now however expressed in terms of the new distributions h_i.

2.6.3 Smagorinsky subgrid model

The idea behind the Smagorinsky subgrid model [129] is to represent the fluid motion at unresolved scales through an eddy viscosity term ν_t depending on the magnitude of the filtered strain rate tensor $\tau_{\alpha\beta}/2\mu$ and added to the normal fluid viscosity ν_0, yielding total viscosity as $\nu = \nu_0 + \nu_t$. The filtering operation refers to a weighted spatial averaging of the form:

$$\bar{w}(x) = \int w(x)G(x, x')dx' \tag{2.58}$$

where w is a fluid quantity of interest, and G is a spatial filter. For finite difference methods, a box filter is used as G. Applying this idea to LBM [57], one can use Eq. (2.58) to define a filtered PDF, and then proceed to define a corresponding filtered LBE. In the remainder of this section, all fluid variables and PDF will be understood to have undergone this filtering operation.

To implement the Smagorinsky model in LBM, the original relaxation time τ_0 is modified

through the addition of a term related to eddy viscosity $\tau_t = 3\nu_t$ with

$$\nu_t = \frac{3}{2}\left(\sqrt{\tau_0^2 + 4c_s^{-4}C_S^2(\Pi_{\alpha\beta}^{\mathrm{neq}}\Pi_{\alpha\beta}^{\mathrm{neq}})^{1/2}} - \tau_0\right) \tag{2.59}$$

where C_S is the Smagorinsky constant, which for LB models can be taken to be 0.1. This effectively abandons the single relaxation time approximation by making relaxation time a spatially and temporally varying quantity dependent on the local gradients of the fluid velocity.

3 Sailfish: high-performance LBM simulations on GPUs

To make the simulations described in the following chapters of this thesis possible without the use of large computing clusters, a highly optimized lattice Boltzmann code called *Sailfish* was developed. The code is released as open source under the LGPLv3 license in the hopes that it will be useful to other researchers, as well as an effort to improve the reproducibility of the results presented here — an aspect of computational sciences that sadly often does not receive enough attention in our opinion.

3.1 GPU architecture and history overview

The development of Sailfish coincided with the rapid rise in popularity of the General-Purpose Computing on GPUs (GPGPU) programming techniques. Sailfish was designed from the ground up to take advantage of these and as such, targets the GPU as its primary computer architecture.

GPUs are specialized pieces of hardware, originally developed to accelerate computationally intensive operations in computer graphics. Throughout the first decade of the 21st century, they became a common component in consumer-grade computers, steadily decreasing in price and increasing in capabilities and the delivered computational power.

This trend did not go unnoticed in the science and engineering communities, which started experimenting with using this new and relatively cheap source of computational power for simulations and other types of numerical computations. This effort was greatly accelerated with the release of the CUDA platform by NVIDIA and the OpenCL standard by the Khronos group. CUDA C and OpenCL, two programming languages structurally very similar to each other, provided the programmer with access to the GPU hardware while hiding many graphics-specific details, thus significantly simplifying the software development effort. In what follows, we will restrict the discussion to CUDA-compatible devices, as this was the hardware architecture on which all the simulations presented in this thesis were performed.

In the context of LBM, GPUs were first used for simulating simple two-dimensional [139] and three-dimensional flows [140] shortly after the initial release of the CUDA platform in 2007. More recently, GPUs have also been applied to more complex, two-dimensional multicomponent flows [15]. To the best of our knowledge, Sailfish is the first attempt to

implement a wide range of models within a single GPU software package, and for some of the discussed models such as ELBM or Free Energy (FE), the first attempt to use them on a GPU in general.

3.1.1 Hardware overview

Modern GPUs are massively parallel computational devices, achieving performance in the range of TFLOPS. CUDA devices can be grouped into four generations, called Tesla, Fermi, Kepler and Maxwell, with each new generation offering better performance, computational efficiency (performance per watt) and additional advanced features (e.g. the shuffle operation in Kepler devices).

The core physical computational unit of a GPU is the Streaming Multiprocessor (SM), itself consisting of several Scalar Processors (SPs). Each SM executes the program code in a Single Instruction, Multiple Threads (SIMT) manner, with one computational thread assigned to every SP and each thread executing the same code, but on different data units. The SM also provides a limited amount of specialized on-chip memory: a set of registers, a shared memory block and L1 cache, a constant cache and a texture cache. The registers are the only type of memory logically local to a SP — the other ones are a shared resource and allow data sharing between threads. It should be noted that this architecture does not provide any facilities for communication between SMs.

In addition to the SMs, the GPU is typically equipped with several gigabytes of Random Access Memory (RAM), known as *global memory*, which is physically located on the graphics card and is distinct from the RAM of the host system.

The various types of memory available to GPU programs can be organized into a hierarchy ordered by the speed of access. The fastest type of memory available to a GPU program is the register and shared memory bank — located physically close to the SM, and limited in size to tens of kilobytes, it offers bandwidth of the order of 1.3 TB/s. The next memory type is the previously mentioned global memory, allowing for data transfers at a rate of about 200 GB/s. All other memory types, such as the host RAM, can only be accessed through the PCI-e link, which limits the available bandwidth to a theoretical maximum of 16 GB/s.

This hierarchical organization of memory is typical for modern computing systems, where more and more of the available silicon is dedicated to data transfers, as opposed to pure computation [94]. The bandwidth provided by GPUs is very competitive in comparison to CPUs. This is particularly important for LB codes, which process and store relatively many bytes per lattice node (e.g. 92 bytes for D3Q19 in single precision, counting the distribution function, density, and velocity).

3.1.2 Software overview

From the programmer's point of view, the GPU hardware can be utilized by running so-called *kernel* functions, which are typically written in CUDA C — a language based on C++, restricting the use of some complex features and adding a few new GPU-specific keywords. A kernel executes the same code in multiple threads in parallel. Every thread can be processing a different piece of data, but they all have to execute the same instructions.

The threads are logically organized into regular 1-, 2- or 3-dimensional blocks. Every thread gets to know its position in the block through the `threadIdx` structure. Blocks themselves are organized into a 1- or 2-dimensional grid (not to be confused with the LB lattice), the position in which is reported via the `blockIdx` structure. Within a block, kernels are organized into *warps* of 32 threads. The warp assignment is based on the natural linear index (with x being the fastest changing coordinate) induced by the thread position within the block.

Kernels are allowed to use conditional instructions such as `if` or `switch`, but these may cause the execution to diverge into multiple branches. When this happens inside a single warp, the different branches are automatically serialized (transparently to the programmer). Thread blocks are mapped to SMs on the GPU, and as such share its limited resources (e.g. shared memory). The programmer retains control on how threads are organized into blocks and grids, so optimization of the usage of these resources can be performed when writing programs.

3.2 Sailfish code design

3.2.1 Guiding principles

Scientific software is typically developed in two very distinct modes. In the *prototype* mode relatively little attention is paid to the performance and scalability of the program, and emphasis is put on the ability to quickly experiment with different approaches while exerting relatively little effort on the programmer's part. Prototype/research code is typically implemented in high-level languages (e.g. Python) or in specialized environments (e.g. Matlab). In contrast, *production* code is highly optimized, and often designed for large-scale computations, e.g. in a cluster environment. Production codes are typically implemented in lower-level, compiled languages such as C++ or Fortran. This process can take many man-years of effort, especially for large simulation packages such as Fluent (commercial) or OpenFOAM (open source).

The Sailfish project was started as an experiment to find a middle ground between these two extremes. The goal was to build a complex simulation package that would not

compromise on performance and scalability, ideally offering state of the art performance for the core models. At the same time, we strived to keep the code flexible to enable fast experimentation.

The main factor enabling the realization of this goal was the new possibilities offered by GPGPU. In fact, Sailfish was designed from the ground up around the general computational model of CUDA and OpenCL. When the project was starting in late 2009, it was less clear than presently that either of these two programming models would be successful, so enough flexibility was built-in into the original design that the computational model could be changed with relatively little effort. This has not so far been necessary, but the option nevertheless remains.

GPGPU introduces a natural split between the GPU (high performance, needs to be programmed separately), and the host (where the main program runs, and where performance is no longer the primary issue). We decided to take advantage of this and use a very expressible, easy to learn, and high level programming language (Python) for the host part, handling the setup of the simulation, simulation control, communication and input/output. The maturity of the language, as well as the availability of a large number of libraries (particularly NumPy and SciPy), bindings with lower-level system libraries (PyCUDA, PyOpenCL), and a computer algebra system (SymPy) were the main factors motivating our choice of the language.

3.2.2 Run-Time Code Generation

In Sailfish, access to the computational resources of the GPU is provided through the aforementioned libraries (PyCUDA, PyOpenCL). The CUDA C / OpenCL code is built automatically using a template-based code generation module (using the Mako template language) and a computer algebra system. This Run-Time Code Generation (RTCG) approach provides a number of advantages, most importantly:

- isolation from low-level details of the underlying hardware,
- portability (OpenCL vs CUDA C supported by a common code base),
- ability to auto-tune the code to select optimal parameters.

Listing 3.1: Full-way bounce-back rule. Left column: source code in Mako. Right column: automatically generated CUDA C code for the D2Q9 lattice.

```
${device_func} inline void ↩
    bounce_back(Dist *fi)
    float t;

    %for i in sym.bb_swap_pairs(grid):
        <% a = grid.idx_name[i]
           opp_i = grid.idx_opposite[i]
           b = grid.idx_name[opp_i] %>
        t = fi->${a};
        fi->${a} = fi->${b};
        fi->${b} = t;
    %endfor
}
```

```
__device__ inline void bounce_back(Dist ↩
    * fi)
{
    float t;
    t = fi->fE;
    fi->fE = fi->fW;
    fi->fW = t;
    t = fi->fN;
    fi->fN = fi->fS;
    fi->fS = t;
    t = fi->fNE;
    fi->fNE = fi->fSW;
    fi->fSW = t;
    t = fi->fNW;
    fi->fNW = fi->fSE;
    fi->fSE = t;
}
```

The RTCG idea is illustrated in Listing 3.1 on the example of the bounce-back rule. Note that a single function in Mako handles arbitrary LB lattices without any code changes.

Several alternatives to RTCG were considered, most notably a domain-specific language (see e.g. [53]) and C++ templates (see e.g. [42, 112, 122]). We deemed the former to introduce too much overhead and an unnecessarily steep learning curve for users. The latter was decided against due to general complexity making the code difficult to learn and debug, as well as lack of support in OpenCL. A large disadvantage was also the impossibility of saving the code generated from the templates for manual inspection and modification — something our RTCG solution readily supports, and which can be very helpful during development and debugging, as well for auto-tuning purposes. The application of RTCG with GPUs to various branches of science and engineering is discussed in [76].

On top of Mako, Sailfish uses SymPy to express a large number of LB models directly using mathematical formulas, which are then automatically transformed into lower-level code. This makes the code more readable (shorter, less repetitive), and easy to verify at the level of mathematics (e.g. the equilibrium distribution function can be easily checked to yield the correct hydrodynamic variables when its moments are calculated). Additionally, it is also possible to automatically optimize the mathematical expressions for performance or precision. The first goal is now largely handled at the level of the compiler. The latter however is unlikely to be handled this way as some model-specific knowledge is necessary for such optimization (e.g. about the relative scales of various terms in the expression in order to minimize round-off errors).

Listing 3.2: The BGK equilibrium function defined using SymPy expressions. `S` is a class with predefined symbols to be used in formulas. `grid` is a class representing the lattice used for the simulation with discrete velocity vectors available in symbolic form in the `grid.basis` list enabling e.g. simple computation of dot products.

```
from sympy import Rational
def bgk_equilibrium(grid):
    out = []
    for ei, weight in zip(grid.basis, grid.weights):
        out.append(weight * (S.rho + S.rho * (3 * ei.dot(grid.v) + Rational(9, 2) * ↩
            (ei.dot(grid.v))**2 - Rational(3, 2) * grid.v.dot(grid.v))))

    return out
```

The use of SymPy is illustrated in Listing 3.2. Note that the equilibrium function is defined using a single line of code, yet can handle any LB lattice that supports the LBGK relaxation model. Since the function is available as a Python object, it can be further inspected in an interactive environment such as IPython, and e.g. used to render the corresponding formula with LaTeX, using SymPy's translation capabilities. Note also that the numerical coefficients in the formula are stored in symbolic form as rational expressions (e.g. `Rational(3, 2)`) instead of floating point values. This makes them free from round-off errors and allows further symbolic simplification.

3.2.3 Simulation execution and control

Sailfish simulations are defined by implementing two classes – one to specify geometry, boundary and initial conditions, and another one to specify the LB model to use, as well as default simulation parameters (see Listing 3.3 for an example simulation script). These parameters can also be overridden from the command line when the simulation is actually started. This object-oriented design has been adopted in order to give the code a modular organization and to maximize code reuse between different simulations.

Listing 3.3: 2D Lid-driven cavity example in Sailfish.

```
from sailfish.subdomain import Subdomain2D
from sailfish.node_type import NTFullBBWall, NTZouHeVelocity
from sailfish.controller import LBSimulationController
from sailfish.lb_single import LBFluidSim

class LDCSubdomain(Subdomain2D):
    max_v = 0.1

    def boundary_conditions(self, hx, hy):
        wall_map = (hx == self.gx-1) | (hx == 0) | (hy == 0)
        self.set_node((hy == self.gy-1) & (hx > 0) & (hx < self.gx-1), ↩
            NTZouHeVelocity((self.max_v, 0.0)))
        self.set_node(wall_map, NTFullBBWall)

    def initial_conditions(self, sim, hx, hy):
        sim.rho[:] = 1.0
        sim.vx[hy == self.gy-1] = self.max_v

class LDCSim(LBFluidSim):
    subdomain = LDCSubdomain

if __name__ == '__main__':
    LBSimulationController(LDCSim).run()
```

Sailfish simulations can be run using one or more GPUs, located physically on one or more machines, connected through a network. If a distributed simulation is desired, the simulation domain is split into cuboid subdomains. The subdomains do not need to provide a dense cover of the simulation bounding box — exploiting this fact can provide significant memory savings in case of irregular geometries.

The control flow for a distributed simulation is illustrated in Figure 3.1. First, a *controller* process is started. This process analyzes the specified configuration to see how many subdomains should be used in the simulation, and among how many machines and GPUs they should be distributed. Once all subdomains are assigned to a GPU, the controller starts a single *master* process on every physical machine, which in turn spawns children processes called *subdomain runners* — one per subdomain. Communication between masters and subdomain runners is maintained using the ZeroMQ library, whereas the controller uses the execnet library to communicate with the masters. The latter library is used because it makes it possible to easily start a process on a remote machine with very little prior configuration.

The subdomain runners perform an initialization procedure, which consists of the instantiation of the geometry and simulation classes, and macroscopic field initialization on the host (in the form on NumPy arrays). The initialized fields are then copied into GPU memory, where they are used to initialize the distribution functions through a custom GPU kernel. A self-consistent initialization step can be optionally performed, which makes it possible to use a known velocity field to compute the density field [100]. After the initialization phase, every runner enters the main simulation loop, executing kernels

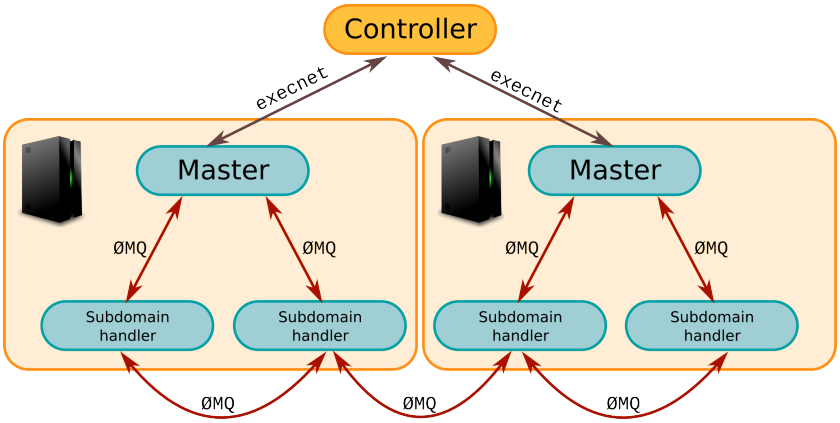


Figure 3.1: High-level architecture of a distributed Sailfish simulation. The simulation begins at the top node representing the controller process. This process starts machine master processes on remote nodes using the execnet Python library. The machine masters in turn spawn children „subdomain runner" processes to handle individual subdomains. Beyond the connections maintained by execnet, communication is facilitated by the ZeroMQ library. In the scheme shown above, the simulation is divided into four subdomains, which are executed on two machines.

implementing the LB algorithm, and exchanging data with the neighboring subdomains through peer-to-peer connections.

For simplicity, the same control flow is used regardless of whether the simulation is truly distributed or not. This does not introduce much overhead, as the role of the master and controller processes is just supervisory beyond the initialization phase. A special debugging mode exists for the case of a simulation with a single subdomain, which makes it possible to run all components (controller, master, runner) in a single process, making Sailfish compatible with standard Python debugging tools.

Subprocesses are used instead of more lightweight multitasking solutions such as threads due to the limitations of the Python runtime in this area (i.e. the existence of a global interpreter lock, limiting parallelism) as well as to simplify GPU resource management in the code.

3.2.4 Core LB algorithm

The key aspects that have to be optimized in order for a GPU program to be efficient, are memory throughput, register utilization, and latency hiding through overlapping of data transfers and computation. Below we discuss how various aspects of Sailfish design address all these.

Memory throughput is determined to a large degree by the underlying data structures. In the case of a LB code, the main data structure is the PDF, for which there are two natural ways to store it in memory — as an array of structures (AoS), or a structure of arrays (SoA). While AoS could be preferred from a software engineering standpoint as it allows for better isolation of high level code from the low-level storage details, it does not fit well with the GPU memory architecture. As such, in Sailfish we opted for a SoA representation in the form of row-major 4D arrays shaped as q, z, y, x. Macroscopic fields are stored in 3D global arrays with the same physical organization.

Global memory reads on a GPU are serviced through atomic transactions of 32, 64 or 128 bytes. On Fermi-class hardware, all memory accesses going through the L1 cache result in 128-byte transactions, while those going through the L2 cache use 32-byte transactions. In order to maximize throughput, one has to ensure that as many bytes as possible in every transaction represent useful data that is going to be used in the code.

A basic Sailfish simulation calls a single kernel called `CollideAndPropagte` repeatedly in a loop, one call per simulation step. This kernel is responsible for collision, propagation, and boundary condition handling. All kernels in Sailfish are organized into 1D blocks of adjustable size. Every node of the simulation domain is handled by precisely one thread.

With the SoA memory organization discussed previously, this results in q read requests to load the PDF from global memory at every LB node. These reads result in contiguous blocks of memory, spanning nodes in the X dimension, and generate fully utilized transactions regardless of the actual transaction size (except at the high X subdomain boundary). Transactions also require naturally aligned memory locations (the base address is a multiple of the transaction size), which are ensured through padding the distribution arrays if necessary.

The propagation step of the LB algorithm shifts distributions by ± 1 in all directions. When this shift takes place along the X axis, the following memory write would be misaligned and could lead to the transaction being split into two. To avoid this, shared memory is used to shift data back to the aligned position within a CUDA thread block. Across block boundaries populations are written directly into global memory, so this problem cannot be completely eliminated.

Earlier works [108, 110] indicate that the cost of unaligned writes is significantly higher than the cost of unaligned reads, and that a propagate-on-read strategy results in up to 15% performance gain. Our experiments confirmed this on older GT200 hardware (GTX 285), however we failed to replicate this effect on Fermi and Kepler devices (Tesla C2050, K10, K20), where the performance of both implementations was nearly identical (typically, with a few % loss for propagate-on-read). This is most likely caused by hardware improvements in the Fermi and later architectures.

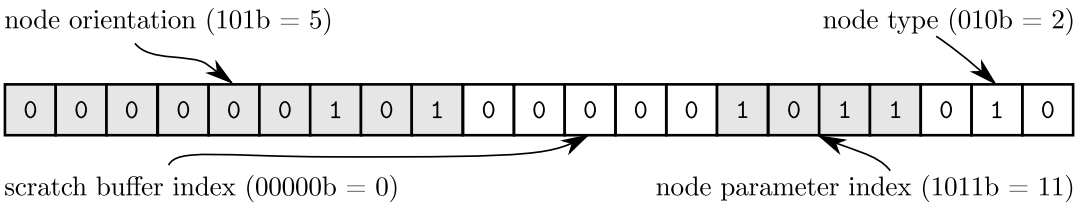


Figure 3.2: Descriptor of a LB node in Sailfish. Every node is represented by such a descriptor in the form of a 32-bit unsigned integer stored in the node type map. The descriptor is logically divided into bitfields (node type, node parameter index, ...) of adjustable size, which is set automatically depending on the needs of the simulation.

3.2.5 Boundary conditions

In addition to the distributions and macroscopic fields arrays, Sailfish keeps an additional array of 32-bit unsigned integers called the *node type map*. Every element of that array is a node descriptor (see Figure 3.2). The node descriptor is formed by variable-size (set at the beginning of the simulation) bitfields, the most important one of which is the node type ID, which indicates what boundary condition should be used at a given node, if any. This ID is followed by a parameter index, which can be used to retrieve any additional boundary condition parameters (e.g. what density to set) from a separate array stored in global memory. Some nodes also use an orientation ID indicating which mass fractions at the node point outside of the simulation domain, as well as a scratch buffer index, which can be used to store temporary data between simulation steps.

All boundary conditions can be made time-dependent by using a SymPy expression or an interpolated time series instead of a single value as their parameters.

3.3 Memory layout

The PDF can be stored in memory in one of two basic patterns, called AB and AA [11]. In the AB pattern, two copies of the PDF are present in memory, called A and B. The simulation alternates between reading from A, and writing to B, and vice versa. In the AA pattern, only a single copy of the PDF is stored in memory, but the logical meaning of a given value stored in memory alternates between simulation steps (see Figure 3.3). The AA pattern is more complex, but leads to a significant 50% memory saving in comparison to the AB pattern. Special care has to be taken to ensure that a given node only reads its own PDF. The PDF of a neighbor cannot be guaranteed not to already

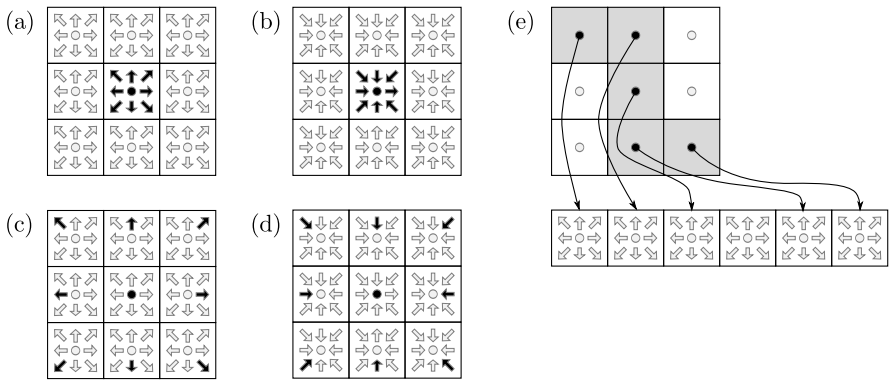


Figure 3.3: Illustration of the three memory layouts supported by Sailfish. In (a)-(d), dark arrows represent distributions used for calculations on the central LB node (middle square in every panel). In the *AB layout*, two copies of the distributions are stored in memory. Data is read from the first copy (a) and after propagation stored in the second copy (c). In the *AA layout*, only a single copy of the distributions exists. Data is read from (a), stored as (b) (same physical location, but different logical meaning, represented by the inverted arrows). In the following step, data is read from (d) and stored as (c). Note that propagation step is now split between the final stage of the first iteration and the initial stage of the second iteration. In *indirect addressing*, pointers from a dense array (one entry per LB node) indicate the physical location of the distributions in a separate continuous array (e). Dark nodes with black circles represent active (fluid) nodes. White nodes represent areas outside of the simulation domain which do not have corresponding distributions.

have been overridden by a value from the following time step. The AA scheme can also be seen to alternate between propagate-on-read and propagate-on-write.

We compared the performance of the AB and AA access patterns on a number of sample simulations. On Tesla-generation devices (GTX 285, Tesla C1060), the AA memory layout results in a slightly higher performance, and is therefore clearly preferred. On newer devices the AB scheme is typically a little faster, but the performance gains over the AA scheme are minor ($< 10\%$), and as such the AB scheme is only preferable when ample GPU memory is available.

In addition to the two basic memory layouts, *indirect addressing* can be used to further reduce memory usage in case of sparse geometries where only a small fraction of nodes from the simulation bounding box represent the fluid. In indirect addressing, every node from the subdomain bounding box is associated with an index into a linear PDF buffer. PDFs are defined only for active nodes (see Figure 3.3), and since PDFs require more storage than the index, this can lead to substantial memory savings, at the cost of some overhead in storage and processing.

In our tests, the use of indirect addressing typically resulted in 5-25% lower simulation performance. This should be taken as a guideline only, as the actual performance impact is necessarily geometry-dependent. The performance is mainly driven by the physical layout of the PDF buffer in relation to the layout of the 3D node array. In an ideal case, 1D blocks from the 3D node space would end up mapped to whole transactions when reading and writing data from the PDF buffer, reducing overall memory access overhead.

3.4 GPU-specific optimizations

3.4.1 Multicomponent models

Multifluid models, such as Shan-Chen or Free Energy, significantly increase the amount of data that has to be transferred between global memory and the multiprocessors. This is driven by two factors — the presence of an additional PDF, as well as the non-local coupling which further increases the number of data reads.

To minimize the impact of the second factor, in Sailfish the simulation step is split into two parts handled by separate kernels. The first kernel computes macroscopic fluid variables and saves them back to global memory. The second kernel performs the collision and propagation step, performing direct reads from global memory whenever nonlocal quantities are necessary, such as in the calculation of the Shan-Chen force or application of a finite difference stencil. While this approach introduces some overhead in that every PDF has to be read from global memory twice (once in each kernel), it significantly reduces the number of reads in the collision/propagation kernel, which

would otherwise need to read distributions directly, which are guaranteed to be more numerous than the macroscopic fields. Without this optimization, it would also be impossible to use the multicomponent models with the AA memory layout.

Throughout the development of Sailfish, we experimented with a number of designs for the collision/propagation kernel:

1. a monolithic kernel handling all fluid components,
2. same as a above, but binding macroscopic fields to GPU textures,
3. a split kernel, called once for every component,

eventually settling on the last one.

The use of textures provided minimal speedups (few percent) on older Tesla devices, which however did not carry over to Fermi and Kepler. Split kernels provided 5.4% (D2Q9, free energy) - 53% (D3Q19, free energy) speedups compared to monolithic kernels. This approach trades off some additional memory reads for lower register usage due to simpler code, leading to higher occupancy of the multiprocessors and hiding the memory overhead. An additional benefit is also the possibility of a straightforward extension of this approach to simulations with more than two components.

3.4.2 Intrinsic functions for ELBM and SC models

CUDA-compatible GPUs contain a special function unit (SFU), which provides hardware implementations of several transcendental functions such as sine, cosine, or the exponential function. The implementation relies on a quadratic polynomial approximation [107] of the target function, and as such is faster, but less precise than its corresponding implementation from the standard math library relying on a Taylor series expansion.

These special capabilities of the SFU are exposed to the programmer through *intrinsic functions* in the CUDA C language. Intrinsic functions have well defined error bounds detailed in the CUDA documentation, though their error can be particularly large when the function argument falls outside of a narrowly defined range. They are also only available in single precision.

We analyzed the performance and precision impact of intrinsic function for the two LB models relying on transcendental functions — the Shan-Chen model with a nonlinear pseudopotential, and the ELBM.

In ELBM, enabling intrinsics as well as the FMAD (Fused Multiply-Add) operation, which allows the calculation of an expression of the form $a + bx$ in a single clock cycle, resulted in a 43% speedup with no visible impact on the correctness of the results in terms of global metrics, such as vorticity or the kinetic energy spectrum. We also noticed that the use of the FTZ (denormalize to 0) mode of the CUDA compiler results in rapid

simulation instability and crash. The stable optimizations can be turned on through the command line arguments `-use_intrinsics -cuda-nvcc-opts='-fmad=true'`. Intrinsics are used for the `logf` function (in entropy calculation), and `powf` (in α estimation and entropic equilibrium). With these optimizations, the overall performance of ELBM reaches 72% of LBGK, making it an interesting alternative for a wide range of problems.

In the Shan-Chen model, the use of intrinsic functions led to 17-20% speedups in both 2D and 3D simulations, with relative changes in density $<$1.5% after 10000 steps.

3.4.3 Shuffle propagation

On lower-end GPUs of compute capability 3.0 or higher, replacing shared memory propagation with the shuffle operation for in-warp data transfers results in a speedup of up to 40%. The shuffle instruction is a new way of sharing data between threads from the same warp introduced by the Kepler GPU architecture. The shuffle operation gives threads direct access to registers belonging to other threads. Using this optimization has the added benefit of freeing up most of the shared memory, which can be used for other algorithms and for the L1 cache. Shared memory is still used for data exchange across warps.

3.4.4 Distributed simulations

The mechanism employed in Sailfish to handle simulations distributed across multiple GPUs is very similar to that used for simulating multiple subdomains on a single GPU. Every subdomain is wrapped by a 1-node thick *ghost node* layer. Ghost nodes are not active (do not undergo relaxation), but are used for data storage and exchange with neighboring layers. For instance, when a particle population is propagated outside of a subdomain through its boundary, it is first saved in a ghost node. An additional CUDA kernel is then executed to move data from the ghost layer into a linear buffer. The buffer is subsequently copied to the host, and sent to remote nodes over a ZeroMQ connection. On the remote nodes, a similar kernel first copies data from the host to the global GPU memory, from which the data is then placed directly in the PDF array according to standard propagation rules.

The ZeroMQ connection may in the future be changed to one based on MPI in order to speed up this part of the simulation through latency reduction and taking advantage of direct data transfer between GPUs and the network card.

To optimize intra-subdomain data transfers, subdomains are divided into two parts called *bulk* and *boundary*. The boundary part is defined as all nodes in CUDA blocks where at least one node touches the subdomain boundary. The simulation step first proceeds through two calls to the `CollideAndPropagate` kernel — one for the boundary region,

and another one for the bulk. As soon as data from the boundary region is available, it can be extracted to the host and exchanged with other subdomains. The data exchange process can run simultaneously with the rest of the simulation step handling the bulk of the subdomain, which effectively hides most of the latency associated with network communication.

As an additional optimization, the exchanged data is limited to the necessary minimum (e.g. only mass fractions that are actually needed in the other subdomain are sent, as opposed to sending the whole PDF). It can also optionally be compressed, which can provide a performance boost when a slow network connection is used, or when only a small fraction of the nodes are active.

3.5 Performance benchmarks

This section presents an overview of the performance of the various LB models implemented within Sailfish. All tests were run using CUDA Toolkit 5.0 and PyCUDA 2013.1.1 on 64-bit Linux systems. Performance of LB models is typically measured in Millions of Lattice site Updates per Second (MLUPS) — a convention we adhere to below.

For single fluid models, the test case used was lid-driven cavity with a 254^3 lattice at $\mathrm{Re} = 1000$. The lid-driven cavity geometry consists of a cubic cavity with one open face (the lid). The open face moves tangentially to the cube with a constant velocity. In the test simulation, half-way bounce-back was used for the walls of the cavity, and the equilibrium velocity condition was used for the lid.

Binary fluid models used a simple spinodal decomposition test case, where a uniform mix of both fluid components fills the whole domain ($254 \times 100 \times 100$) at the start of the simulation. The simulation had periodic boundary conditions enabled in all directions.

The sizes of the simulation domain were chosen to fill a large fraction of the GPU memory (see Section 3.6.2 for why this is important), and at the same time not require adjustment between different GPU models and test configurations (block size, memory access pattern). Whenever such an adjustment was required, the size of the domain in the Z direction was reduced until it fit in memory.

3.5.1 Performance analysis

Comparison of the data from Table 3.1 with results published in the literature [46] shows that Sailfish delivers state of the art performance, in some cases exceeding previously reported figures. To the best of our knowledge, the results presented in this section are the only currently available comprehensive performance comparison of various LB

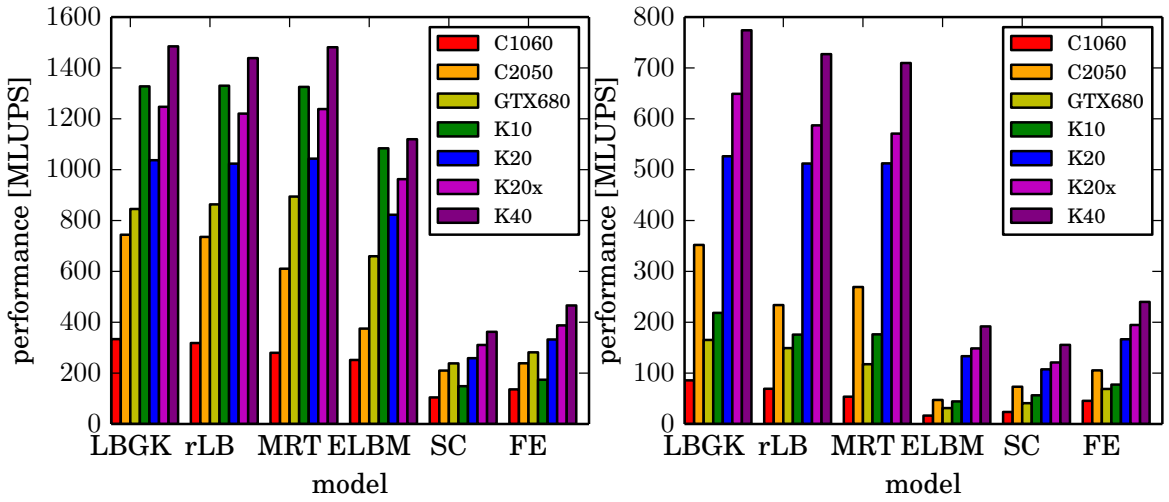
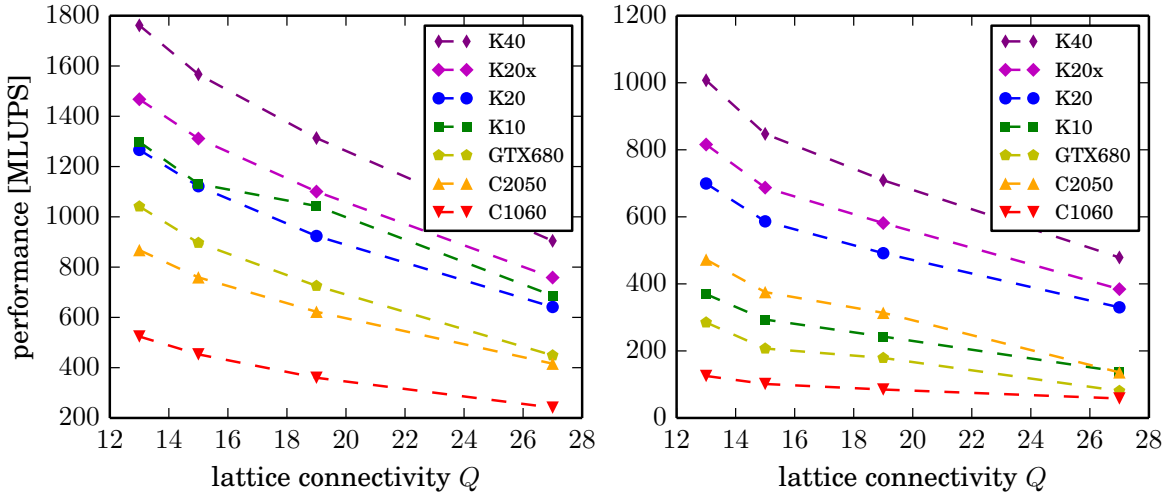


Figure 3.4: Performance comparison of different models using the D3Q19 lattice with the AB memory access pattern. All GPUs had ECC disabled (where applicable). ELBM used intrinsic functions as described in Section 3.4.2. Both logical GPUs were used for the K10 tests. The SM clock of the K40 card was set to 875 MHz. Test case: 3D lid-driven cavity. See Table 3.1 for numerical values. Left panel: single precision. Right panel: double precision.

Table 3.1: Performance comparison of different models using the D3Q19 lattice and AB memory access pattern.

LB model	single precision performance [MLUPS]						
	C1060	C2050	GTX680	K10	K20	K20x	K40
LBGK	333	744	845	1328	1037	1247	1484
RLB	318	736	864	1330	1024	1220	1439
MRT	280	611	894	1325	1043	1238	1481
ELBM	252	375	660	1084	823	963	1120
Shan-Chen	104	210	238	148	258	310	363
free energy	136	238	281	174	322	387	466
	double precision performance [MLUPS]						
LBGK	85	352	165	218	526	649	774
RLB	69	234	149	176	512	587	727
MRT	53	269	118	176	512	571	710
ELBM	16	47	31	44	133	149	192
Shan-Chen	24	73	41	56	107	121	156
free energy	46	105	69	77	166	194	240

Figure 3.5: Performance of an LBGK simulation with the AA memory access pattern as a function of lattice type. The GPUs had ECC disabled (where applicable). Both logical GPUs were used for the K10 tests. The SM clock of the K40 card was set to 875 MHz. Test case: Kida vortex (see Section 6.2). Left panel: single precision. Right panel: double precision.



models on the GPU platform.

Analysis of the test results shows that single precision simulations run at $\sim 80\%$ of the theoretical bandwidth as reported in NVIDIA whitepapers, and close to 100% of the real bandwidth measured by sample code from the NVIDIA SDK. As such, the performance of the code can be considered limited by global memory bandwidth.

Double precision simulations on the other hand run at $\sim 60 - 80\%$ of the theoretical maximum. On older GPUs (Fermi class) they are limited by double precision instruction throughput. On Kepler hardware, they become bandwidth-limited, similarly to single precision simulations.

Overall, memory bandwidth can be seen to be a reliable indicator of expected performance for single precision simulations. This is shown by both measuring the memory bandwidth usage empirically, and by the fact that in Figure 3.5 simulation performance is inversely proportional to the lattice connectivity q in single precision. Planned advances in GPU technology, such as stacked DRAM (scheduled for the Pascal architecture, which is to follow the currently available Maxwell), promise significant progress in terms of memory bandwidth. If realized, they should translate into direct gains in LB simulation performance.

Table 3.1 shows that performance of the simulation goes down with the increasing com-

plexity of the model. In all cases, LBGK being computationally the simplest collision operator, offers the best performance. RLB and MRT perform similarly well in single precision, but in double precision a larger performance drop can be observed, which can be attributed to higher register usage. Remarkably, with the implemented optimizations, ELBM, sometimes considered to be significantly more expensive than LBGK, is only 25% slower. This is largely attributable to the use of intrinsic functions, as the performance drops considerably in double precision (to only 25% of LBGK).

Binary fluid models are significantly slower than single fluid ones, which is understandable in the context of LB simulations being bandwidth constrained. Between the Shan-Chen and free-energy models, the latter is 30-50% faster, with the performance gap being larger in double precision. This can be explained by the lower number of global memory reads/writes that the free-energy model requires in comparison to the Shan-Chen model.

Figure 3.4 and Figure 3.5 clearly show that newer GPUs perform better (as could be expected) with the K40 card delivering the highest performance of all test systems. The K10 GPU provides an interesting solution that works very well in single precision with speeds in excess of 1.3 GLUPS per board in D3Q19 (in Figure 3.5 the lower performance for the K10 test of D3Q13 and D3Q15 lattices is caused by the overhead of copying data between the two physical GPUs through the host).

3.5.2 Optimization strategies

On Fermi hardware, increasing the L1 cache size to 48 kB, disabling the L1 cache for global memory accesses, and replacing division operations by the equivalent multiplication operations where possible has a large positive impact on code performance (~ 3.5 speedup in total).

The L1 optimization strategy does not seem to apply to Kepler-class devices. This is likely caused by the fact that double precision code uses more registers, of which fewer are available on Fermi devices. Freeing up the L1 cache makes register spilling more efficient (accesses to registers spilled to local memory always go through the L1 cache), as fewer operations require communication with global memory.

3.6 Multi-GPU scaling

While the performance offered by a single top-of-the-line GPU is impressive (see Table 3.1), the size of the simulation is still limited by the global memory available on the GPU, which is of the order of gigabytes. This brings about the necessity of distributed simulations, where more computational resources can be used at the same time. If the simulation software has good scaling properties, the size of the problems becomes limited only by the available hardware.

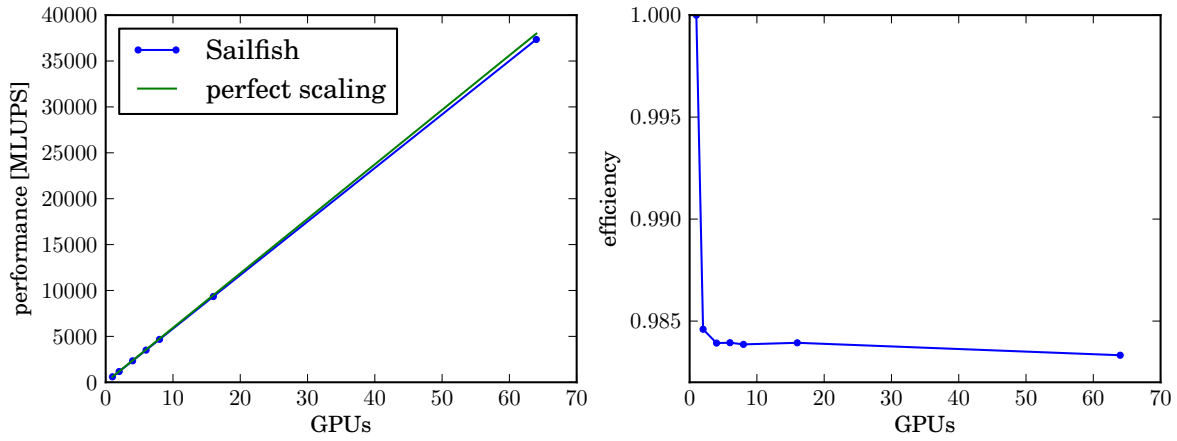


Figure 3.6: Weak scaling properties of the Sailfish code on the 3D duct flow test case. Left panel: absolute performance values. Right panel: efficiency fraction.

The subsections below present the results of performance tests of distributed simulations on the Zeus cluster, which is part of the PLGRID infrastructure. The cluster consists of nodes with 8 M2090 GPUs, connected with an Infiniband QDR network. The simulation used for the tests was a 3D duct flow geometry, with periodic boundary conditions in the streamwise Z direction and half-way bounce-back walls at the remaining boundaries.

3.6.1 Weak scaling

Weak scaling measures code performance when computational resources are allocated in proportion to the size of the problem. In our case, we scaled the domain size $254 \times 127 \times 512 \cdot N$, where N is the number of GPUs. The domain was split into N equal-size subdomains, such that one subdomain was assigned to every GPU. The simulation ran in single precision with a CUDA block of 128 threads, and used the D3Q19 lattice with the AA memory organization.

Figure 3.6 shows performance up to 64 GPUs, which was the largest job size possible to run on the cluster at the time the test was performed. Excellent scaling properties can be observed, with the minimal 1.5% efficiency loss present whenever multiple subdomains are used being largely independent of the actual number of subdomains. Additional optimization techniques, such as peer-to-peer memory copies between GPUs located on the same machine could enable further mitigation of any performance losses in this configuration.

The good performance of distributed simulations is enabled by the solutions described in Section 3.4.4, which make it possible to largely hide the impact of network traffic. When the bulk/boundary split was disabled, simulation performance degraded by 4.5%

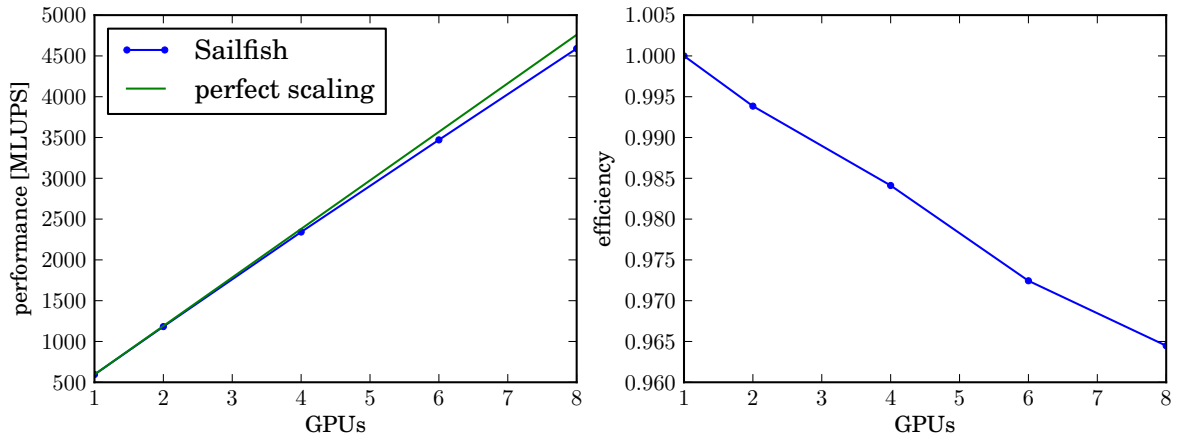


Figure 3.7: Strong scaling properties of the Sailfish code on the 3D duct flow test case. Left panel: absolute performance values. Right panel: efficiency fraction.

for subdomains that fit on the GPUs on a single machine, and by 14% when the use of network connections was required.

3.6.2 Strong scaling

In contrast to weak scaling, in strong scaling tests the size of the computational problem is kept constant as more computational resources are added. We used the same general settings as in our weak scaling tests and set the domain size at $254 \times 127 \times 1664$ (which was the largest that fit within the memory of a single M2090 GPU). The domain was then divided into equal-length subdomains along the streamwise axis, with one subdomain assigned to every GPU used in the test.

Figure 3.7 presents the results of this strong scaling test. The performance is slightly worse than in the case of weak scaling, but even when the simulation is expanded to 8 GPUs, the efficiency loss is still only 3.5%.

For practical purposes, it is important to note that there is a minimum domain size below which performance quickly degrades due to the overhead of using the GPU resources. For our tests, this size can be estimated to be $\sim$ 14% of the GPU memory (or 8.2 M lattice nodes, see Figure 3.8). The exact value is problem-specific, but can be expected to be of the same order of magnitude for simulations using similar models and geometries.

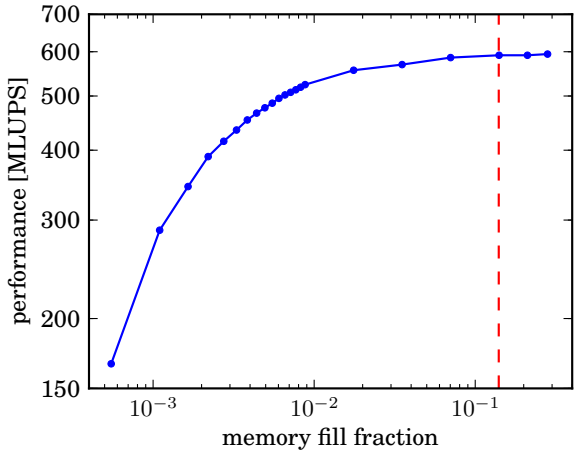


Figure 3.8: Duct flow simulation performance as a function of used memory fraction of a single M2090 GPU (memory use was controlled by varying the extent of the domain in the streamwise direction). The vertical line shows the point at which the computational capabilities of the GPU are saturated.

3.7 Impact of single precision

GPUs, especially those from generations prior to Kepler, are significantly faster for single precision calculations, with the speedup factor varying between 10 for older devices (Tesla class) and 2 for newer ones (Kepler class). In addition, even state of the art GPUs that are not targeting the High Performance Computing (HPC) market, have their double precision capabilities significantly limited, directly reducing performance. Being able to run simulations in single precision is therefore important for practical reasons, as commodity video cards are significantly cheaper than dedicated HPC hardware.

Other authors used the lid-driven cavity benchmark problem to check if single precision models produce good quality results [87, 109]. We decided to test Sailfish on a problem with known analytical solution — the 2D Taylor-Green decaying vortex flow, the state of which at time t is given as:

$$\begin{aligned} u_x(t) &= -u_0 \cos(x) \sin(y) e^{-2\nu t} \\ u_y(t) &= u_0 \sin(x) \cos(y) e^{-2\nu t} \\ \rho(t) &= \frac{\rho}{4} \left(\cos(2x) + \cos(2y)\right) e^{-4\nu t} \end{aligned} \tag{3.1}$$

where u_0 is a velocity constant and $x, y \in [0; 2\pi)$. Periodic boundary conditions are assumed for this system. Eq. (3.1) can be easily verified to be a solution of the incompressible NSEs.

To test the accuracy of our code, we performed multiple runs of the simulation on a 256^2

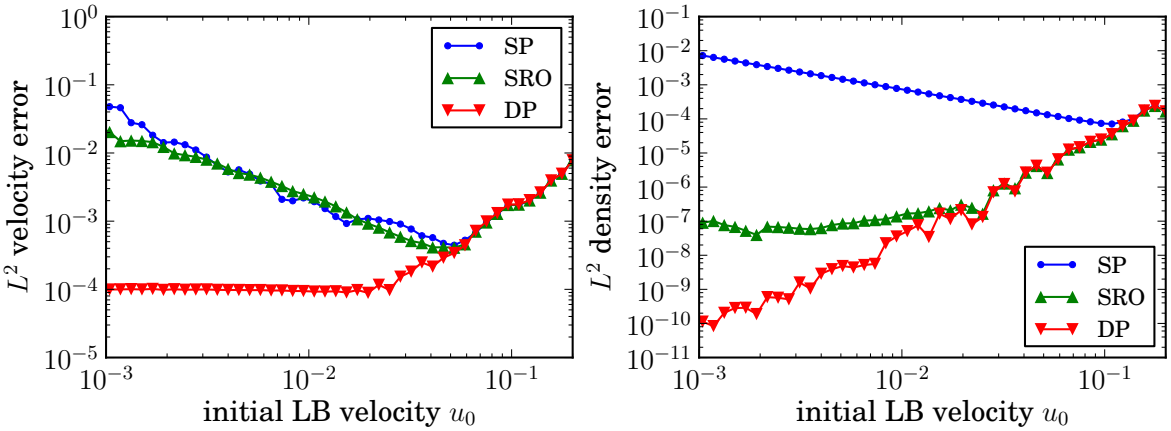


Figure 3.9: Solution error for the Taylor-Green vortex test case with the LBGK model on a D2Q9 lattice, in single precision (SP), single precision with round-off minimization (SRO) and double precision (DP). Left panel: velocity error. Right panel: density error.

D2Q9 lattice with the LBGK collision operator. The numerical viscosity ν was varied between runs, while the Reynolds number remained fixed at Re = 1000. Simulations were run until a specific point in simulated physical time, corresponding to 10^6 steps at $u_0 = 0.0005$. We tested calculations in single precision (SP), double precision (DP) and single precision with round-off minimization (SRO). To quantify the deviation of the numerical solution $\hat{u}_x$, $\hat{u}_y$ from the analytical one (Eq. (3.1)), the relative L^2 error norm for the velocity and density fields was used:

$$\epsilon = \sqrt{\frac{\sum_{\text{nodes}} (u_x - \hat{u}_x)^2 + (u_y - \hat{u}_y)^2}{\sum_{\text{nodes}} u_x^2 + u_y^2}}. \tag{3.2}$$

The left panel in Figure 3.9 shows that in double precision the velocity error stays constant until $u_0 \approx 0.02$, and raises quadratically for higher values, as expected from $\mathcal{O}(\text{Ma}^2)$ accuracy of the LBGK model. In single precision however, lower velocities lead to higher errors. The optimum value of velocity seems to be $u_0 \approx 0.05$ where the interplay of model accuracy and numerical round-off errors causes the velocity error to reach a minimum. For higher velocities, the error is the same as for double precision.

The right panel in Figure 3.9 shows interesting results for the round-off minimization model. While for the velocity field the error for SRO and SP were comparable, SRO clearly performs better in the case of the density field. For both SRO and DP, lower values of maximum velocity reduce the compressibility error. For SP calculations, the optimum value for the density field is $u_0 \approx 0.1$.

Overall, the test illustrates two interesting facts. First, for simulations at higher Mach

numbers, there is often no real difference between single and double precision. Second, single precision calculations have a „sweet spot" at moderate Mach numbers where errors are minimized. For practical purposes, LB simulations are typically run at Ma ≈ 0.1, as lowering the average velocity directly leads to longer simulation times. This puts these simulations in a regime where little difference is seen between SP and DP, making the former preferable for performance reasons. Somewhat counterintuitively, lowering the Mach number will lead to a decrease of accuracy unless DP is used, which puts a limit on how much performance can be traded off for precision without changing the floating point format. Finally, the SRO model should clearly be preferred over SP, as it can only increase simulation precision and does not have a computational overhead that could negatively impact performance.

3.8 Validation

Validation of all the implemented models on a number of representative examples is presented in [64]. Here, we show some additional tests directly relevant to the issues discussed in the following chapters.

3.8.1 No-slip boundary conditions

In order to measure the accuracy of the no-slip boundary conditions, we simulated flows in two- and three-dimensional geometries, for which profiles can be computed analytically. The simulations were conducted at constant Reynolds numbers, but the channel diameter R was varied between runs. This made it possible to estimate the order of convergence for the boundary conditions. In all cases, the pressure gradient driving the flow was replaced by a body force implemented using Guo's method. The simulations were run in double precision at $u_{\max} = 0.02$ and $\mathrm{Re} = Ru_{\max}/\nu = 125$, until convergence of the L^2 error was reached.

In two dimensions, we chose the standard Poiseuille geometry (flow between two parallel no-slip walls):

$$u(r) = \frac{1}{4\mu}\frac{\partial p}{\partial x}(R^2 - r^2) \tag{3.3}$$

where R is diameter of the channel, and r is the radial coordinate, defined as $r = y - R$, assuming (in Cartesian coordinates) a channel infinite in the x direction, and spanning $y \in (-R, R)$. To simulate an infinite channel, we used periodic boundary conditions in the streamwise direction. Since the length of the simulation domain does not matter in such a setup, we used a constant length of 32 nodes.

The test procedure was repeated in three dimensions using the D3Q19 lattice, using the geometry of a channel with a rectangular cross-section of sides a and b. The analytical

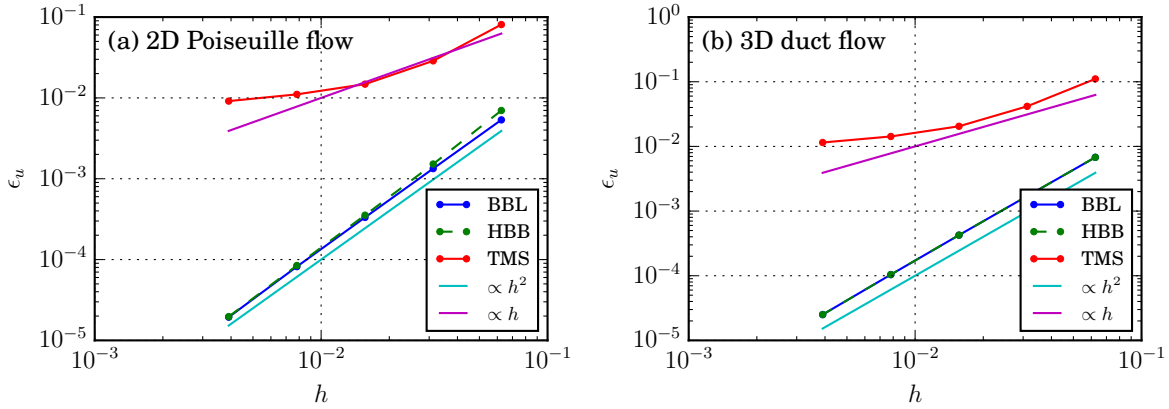


Figure 3.10: L^2 error norm (see Eq. (3.2)) of the numerical solution of (a) the 2D Poiseuille flow test case, and (b) the 3D duct flow test case, as a function of grid resolution. The analytical solution is given by Eq. (3.3), and Eq. (3.4), respectively.

solution for this case is [151]

$$u(y,z) = \frac{16a^2}{\rho\nu\pi^3}\left(-\frac{\partial p}{\partial x}\right)\sum_{i=1,3,5,\ldots}^{\infty}(-1)^{(i-1)/2}\left(1-\frac{\cosh(i\pi z/2a)}{\cosh(i\pi b/2a)}\right)\frac{\cos(i\pi y/2a)}{i^3} \tag{3.4}$$

for $-a \leq y \leq a, -b \leq z \leq b$. The chosen geometry let us keep the boundaries aligned with the Cartesian axes of the lattice, avoiding additional errors introduced by the staircase approximation of a curved boundary.

Figure 3.10 shows that as expected both half-way bounce back (HBB) and bounce-back on the link (BBL) are second order accurate in the spatial resolution h, both in two and three dimensions. The Tamm-Moth-Smith (TMS) method though, exhibits only first order spatial accuracy, and results in a significantly higher absolute error. For the purpose of this analysis, we assumed that the location of the wall is the same as in HBB, that is in the middle of the line connecting the TMS node and the nearest fluid node. In general though, the effective location of the wall (i.e. the lattice coordinate system location where $u = 0$) appears to be a function of relaxation time and maximum velocity, making TMS hard to use in cases where a priori knowledge of wall location is necessary.

3.8.2 Womersley flow

The Womersley flow describes time-dependent, cyclic flow in rigid tubes and was originally created as a model of pulsatile blood flow in arteries [152]. Similarly to Poiseuille

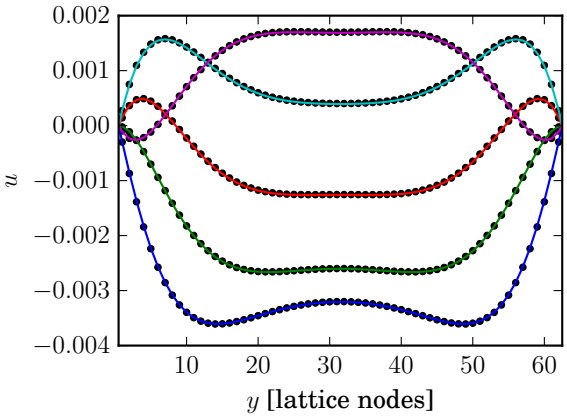


Figure 3.11: Velocity profiles of a 3D Womersley flow simulation at selected time steps. Continuous lines are computed using Eq. (3.6), dots show LBM simulation results.

flow, the Womersley flow is driven by a pressure gradient. The derivation of the time-dependent velocity profile starts with the Poiseuille solution, and then expresses the pressure gradient as a periodic function of a given angular frequency ω:

$$\frac{\partial p}{\partial x} = \sum_{n=-N}^{N} C_n \exp(in\omega t) \tag{3.5}$$

which represents the arterial pulse. The solution can then be shown to have the form:

$$u(r,t) = \mathrm{Re}\left(\sum_{n=-N}^{N} i\frac{C_n}{n\omega\rho}\left[1 - \frac{J_0(i^{3/2}n^{1/2}\alpha(r/R))}{J_0(i^{3/2}n^{1/2}\alpha)}\right]\exp(in\omega t)\right) \tag{3.6}$$

where J_0 is the Bessel function of the first kind and order 0, α is the Womersley number as defined in Chapter 2, r is the radial coordinate (with the channel axis corresponding to 0), and R is the channel radius.

In order to test the implementation of time-dependent boundary conditions, we ran a 3D Womersley flow simulation at $\mathrm{Re} = 124$, $\alpha = 6.93$, $\omega = 5 \cdot 10^{-4}$, $\partial p/\partial x \approx 4.24 \cdot 10^{-4}\sin(\omega t)$ on a $256 \times 64 \times 64$ domain, using the D3Q19 lattice. Figure 3.11 shows good agreement of the obtained results with the theoretical predictions. Note that the streamwise size of the domain does not impact the simulation materially as long as the oscillation period $T = 2\pi/\omega$ in LB units is significantly longer than the length of the channel. When this is not the case, LBM with its finite speed of sound will no longer provide a good approximation of an incompressible fluid.

4 Two-component flows

4.1 Background

In an important class of chemical reaction engineering problems, gas and liquid phases are contacted with a solid catalyst. The physical setup of such systems can take many forms, including:

- *slurry reactors*, where the catalyst is suspended in the liquid, through which the gas is then driven in the form of bubbles,
- *trickle bed reactors*, in which both the gas and the liquid trickle down driven by gravity over a packed bed of catalyst particles,
- *monolith reactors*, which are divided into a number of channels through which the fluid and gas are driven, and the reactions take place primarily on the walls of the channel which are coated with a catalyst.

In what follows, we focus on monolith reactors. These were initially used in car exhaust pipes, but have since found many more applications [13, 45, 80, 159]. They are typically made of ceramic or metallic materials, and have geometrically simple cross-sections — circular, square/rectangular, or triangular. They offer a number of important practical advantages. Higher catalyst surface area allows for faster reactions, and lower fluid pressure drop reduces energy usage. Compared to other methods, they are also easier to clean (simple geometry), and allow for easy scale-up of operations by mass-manufacturing of the reaction channels.

From the fluid dynamical point of view, such monolith reactors can operate in a number of flow regimes, in order of increasing gas flow rate: bubbly flow, Taylor flow, annular flow, and churn (semi-annular) flow.

In the Taylor flow, the catalyst wall is separated from an elongated gas bubble by a very thin film. This makes it possible to attain a high concentration gradient within the film. The Taylor gas-liquid flow pattern is characteristic of small channels, where surface tension forces dominate over gravitational forces.

4.2 Motivation

In this chapter, we study the applicability of the free-energy LB model to the Taylor flow problem, and the associated domain resolution requirements necessary to obtain good results. The free energy model is a diffuse interface method, where the interface between the two phases can span several nodes. This spatial extent of the interface is however an artifact of the method. It is therefore important to set up the simulation in such a way that the interface size does not affect the physics of the thin film separating the wall of the microchannel from the gas bubble. We will focus our discussion of the resolution requirements on this problem.

The problem of Taylor flow in microchannels has primarily been studied using volume of fluid [92], finite element [48, 81], and level-set [38] methods. Within FEM, the flow was treated as a free surface problem with a sharp interface, and the physics of the gas phase was not considered [81]. The use of a diffuse interface method for Taylor flow is interesting, as it makes it possible to relatively easily tackle issues such as droplet coalescence and breakup, as well as unconstrained interface motion and complex boundary geometries.

Within the framework of LBM, Taylor flow was studied using the Shan-Chen model [158]. As we have seen in Chapter 3, the free energy method is computationally more efficient. It is also easier to use, as parameters such as interface tension can be calculated explicitly, whereas in Shan-Chen model (SC) they need to be estimated from Laplace's law simulations.

Gas bubble flows can be characterized through the *Capillary number* expressing the ratio of viscous and surface tension forces, and defined as

$$\mathrm{Ca} = \mu_{\mathrm{liq}} u_{\mathrm{bubble}} / \gamma, \tag{4.1}$$

where γ is the (gas-liquid) interfacial tension.

Bretherton [19] showed that the front meniscus thickness is $\propto \mathrm{Ca}^{2/3}$ for $\mathrm{Ca} < 0.003$. Later studies [153, 154] demonstrated however that this conclusion is valid only in a specific region behind the front meniscus and that interface thickness is not constant along the bubble. The lack of general theoretical models of film thickness motivates the use of numerical simulations for the Taylor problem.

4.3 Free energy model

The free energy model [134] assumes uniform density in both components. This is in general not the case for the Taylor/Bretherton problem, where a gas and a liquid are used. However, prior research [19, 54] suggests that inertial effects play a negligible role.

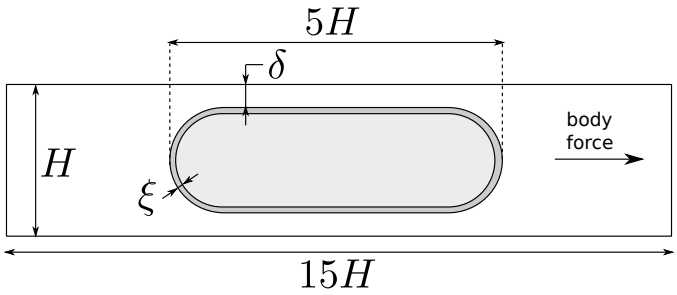


Figure 4.1: Geometry of the 2D microchannel case (not to scale). Periodic boundary conditions are applied in the horizontal direction. δ indicates the physical film width (distance between the bubble and the channel wall), while ξ indicates the size of the diffuse interface.

Giavedoni and Saita [41] showed that $\mathrm{Re} \in (0, 70)$ has a negligible impact on film thickness for $\mathrm{Ca} < 0.05$ and a moderate impact for $\mathrm{Ca} > 0.05$. Heil [54] extended this conclusion up to $\mathrm{Re} = 300$, where a 7% film thickness deviation from that measured at $\mathrm{Re} = 0$ was observed. The Reynolds number has been noted to be important for pressure distribution, and for flow field near the front cap of the bubble.

The density ratio can also impact the bubble shape. Its influence can be characterized through the *Bond number* Bo, which is the ratio of body forces to surface tension forces. In the microchannels considered here, the surface tension forces dominate and $\mathrm{Bo} < 0.1$, so any impact on bubble shape can be neglected.

Overall, we can conclude, that it is the viscosity ratio, and not the density ratio, that is most important for modeling the Taylor flow, which motivates our choice of the free energy model for the simulations.

4.4 Simulation setup

The geometry of the simulation is that of an elongated channel, with half-way bounce-back nodes used for the walls at the top and the bottom of the domain (see Figure 4.1). The simulation domain is periodic in the horizontal direction, modeling a channel of infinite length. The domain is initialized with a uniform field of $\rho = 1$ and $\vec{u} = \vec{0}$. The bubble starts as a rectangular slab of $\phi = -1$, separated from the wall by a liquid film of width proportional to $\mathrm{Ca}^{2/3}$. Outside of the bubble, the whole domain is assumed to be filled with the liquid phase and initialized to $\phi = 1$.

Since we use periodic boundary conditions in the streamwise direction, the actual simulated system is that of a bubble train, not a single bubble. To minimize inter-bubble influence, we limited the length of the bubble to $5H$, and placed it centrally in a channel

of length $15H$. Wong et al. [153] showed that the bubble should be long in order of the film thickness to be proportional to $\mathrm{Ca}^{2/3}$, which motivated our decision to not reduce the bubble length further.

All simulations were started with a target value of the capillary number Ca, which was used to compute the speed of the bubble u_{bubble}. We also kept $\mathrm{Re} = Hu_{\mathrm{bubble}}/\nu < 10$ at all times to minimize the impact of any inertial effects.

4.5 2D simulations

The primary objective of the simulations was to obtain reliable film thickness results. A number of careful studies were conducted to ensure that the measured thickness is not affected by the size of the diffuse interface, spurious currents, wettability coefficient, and grid size.

The simulations used $\kappa = 0.04$ and $a = 0.04$, $\tau_{\mathrm{liq}} = 2.5$ and $\tau_{\mathrm{gas}} = 0.7$, corresponding to a 1:10 viscosity ratio. The capillary number range of $\mathrm{Ca} \in (0.01, 1.0)$ was explored (note that simulation memory requirements grow as Ca decreases — a larger lattice is needed to properly resolve the very thin film). For every target Ca, the corresponding bubble velocity u_{bubble} was computed from the definition of the Capillary number Eq. (4.1). This velocity was used to compute the Poiseuille pressure gradient Eq. (3.3), which in turn was translated to a body force and applied using Guo's body force implementation. Note that since the simulated flow is not actually a Poiseuille flow, and since the bubble velocity differs from that of the surrounding liquid, this is only an approximation and the actual bubble velocity seen in the simulation is expected to deviate from the target value. Whenever this happened when a specific value of Ca was needed, the simulation had to be repeated with an adjusted body force until the expected value was reached.

When analyzing the simulation data, the film width δ was always measured in the middle of the bubble. As the interface between the two phases in the free energy model is symmetric, we took the point where $\phi = 0$ as the location of the bubble interface. To find this location, a 3rd order spline was fit to the phase profile data. The root of the interpolating spline was then identified using the Brent routine. This procedure was used to determine the bubble tip and end, as well as its width in the middle. The bubble velocity was computed as:

$$u_{\mathrm{bubble}} = \frac{x_{\mathrm{mid}}(t_1) - x_{\mathrm{mid}}(t_0)}{t_1 - t_0}$$

where $x_{\mathrm{mid}}(t_i)$ is the location of the bubble midpoint at step t_i of the simulation. A large enough $\Delta_t = t_1 - t_0$ was used for the estimate (typically, $\Delta_t = 2.5 \cdot 10^4$) and all measurements were made once the system reached equilibrium.

The width of the diffuse interface ξ was estimated using the same procedure as that used

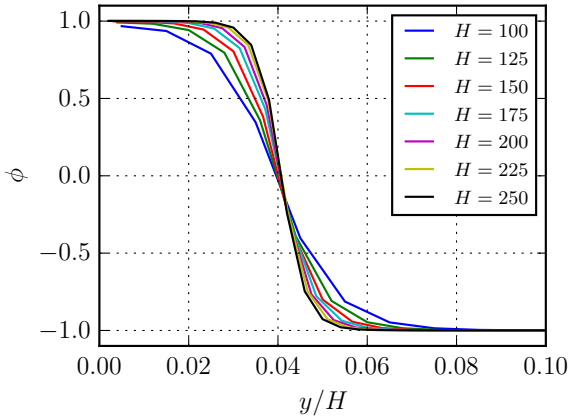


Figure 4.2: Impact of the lattice size on the interface profile (and thus film thickness) measured in the middle of the bubble at an effective capillary number Ca $\approx 10^{-2}$.

for δ, taking ± 0.9 as limiting points of the interface. ξ depends on the model parameters a and κ and not on grid size. With the parameters used in our simulations, $\xi \approx 4.07$.

4.5.1 Grid independence

In order to evaluate how large the simulation domain needs to be to properly resolve the thin film around the bubble, we ran multiple simulations keeping the capillary number constant at Ca $\approx 10^{-2}$ while gradually increasing the lattice size $15H \times H$ from $H = 100$ to $H = 250$.

The resulting phase field profiles are presented in channel height-normalized form in Figure 4.2. At $H = 200$, the film thickness is within 2% of its target value (taken from the highest resolution simulation at $H = 250$). With the film occupying $\delta \approx 10.18$ nodes (on one side of the bubble) at $H = 250$, we estimate that the film should be resolved at approximately 2.5 the size of the diffuse interface ξ.

4.5.2 Wall wettability

In the free energy model, wall wettability is controlled through the gradient of the phase field ϕ at the no-slip nodes [118]. In order to check whether it impacts film thickness results we ran simulations at a relatively high resolution of $H = 200$ until convergence for various values of wall-normal gradient of the phase field $\partial_n \phi$, implemented through a first order finite difference scheme. Figure 4.3 shows that film thickness is not affected (differences are less than 0.2%). It should be noted that negative gradient values cause $\phi > 1$ close to the wall — an unphysical value which goes beyond the model range

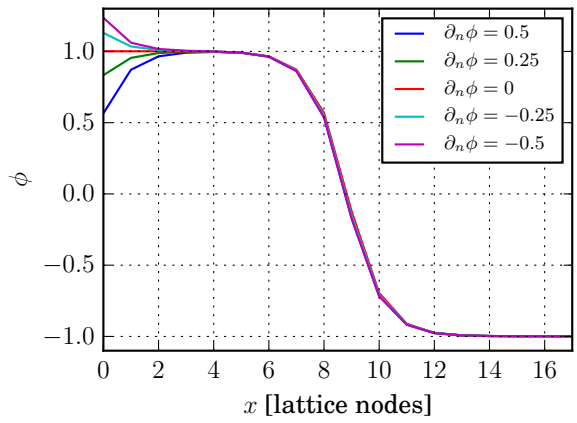


Figure 4.3: Size of the film width after $5 \cdot 10^5$ iterations for different values of the wetting angle. The channel height was set to $H = 200$, the initial film width was 10 nodes, and the desired capillary number was $\mathrm{Ca} = 0.1$.

of $\phi \in [-1, 1]$. Values of the gradient larger than those shown in Figure 4.3 do affect simulation results. $\partial_n\phi = -0.75$ caused the film thickness to expand by ~ 1 node, while with $\partial_n\phi = 0.75$ the bubble started sticking to the wall, changing the flow regime.

4.5.3 Comparison with literature

Giavedoni and Saita [41] performed a detailed study of film thickness δ as a function of Ca and Re, showing significant deviations from the theoretical prediction of $\delta = 1.3375\mathrm{Ca}^{2/3}$ for $\mathrm{Ca} > 10^{-3}$. We compared our results to their data at $\mathrm{Re} = 0$, showing very good agreement (relative differences $< 1\%$) within the range $\mathrm{Ca} \in (10^{-2}, 1)$ (see Figure 4.4). At lower capillary numbers a slight deviation starts to be visible — an indication that the simulation domain is not large enough to properly resolve the film.

4.6 3D simulations

In comparison to the 2D case discussed previously, more experimental [47, 137] and numerical [48, 92, 153, 154] data is available for the 3D problem. The higher dimensionality allows some novel interesting phenomena to appear. In rectangular capillaries, a transition from an asymmetric to an axisymmetric bubble shape takes place at a critical capillary number $\mathrm{Ca}_{\mathrm{crit}}$ [48, 153]. The value of $\mathrm{Ca}_{\mathrm{crit}}$ is not known precisely, and different estimates were reported in the literature, ranging from 0.033 [48], 0.04 [137], up to 0.1 [92].

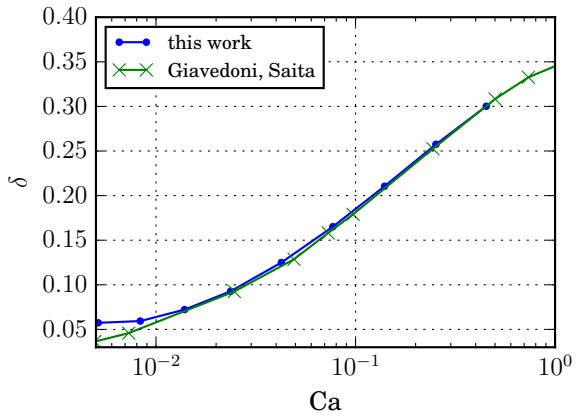


Figure 4.4: Comparison of film thickness data between present work and the results of Giavedoni and Saita [41] (data extracted using the PlotDigitizer software).

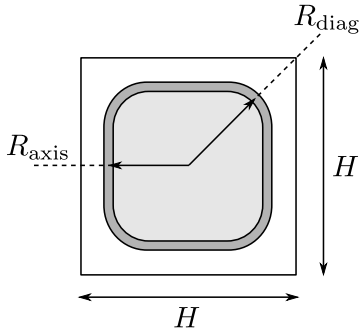


Figure 4.5: Geometry of the 3D microchannel in the plane orthogonal to the flow direction. Geometry in the streamwise direction z remains the same as in Figure 4.1.

Hazel and Heil [48] performed simulations for channels with circular, square and rectangular cross sections. They noted a transition in the flow regime in the liquid slug, where vortices present at lower capillary numbers disappear once $\mathrm{Ca} > 0.691$. Hazel and Heil [48] also found an empirical correlation for the functional dependency of the bubble radii on the capillary number for rectangular channels of different aspect ratios.

In our simulations, we used a geometry that follows our approach established in the 2D case, with the whole domain shaped as $H \times H \times 15H$ and the bubble being H nodes long (see Figure 4.5). Hazel and Heil [48] reported that the bubble radii are approximately $0.49H$, which, if we were to resolve the film with 6 LB nodes, would lead to a very large domain size of $H = 600$. Fortunately, in our simulations the film thickness was actually larger, and the domain sizes we used were limited to $H \in [100, 200]$, which are much easier to manage from the point of view of computational resources.

4.6.1 Implementation

To further reduce the computational requirements, the inherent symmetry of the system can be exploited to decrease the domain size by a factor of 4. In order to only simulate a quarter of the original channel volume, we used symmetric boundary conditions at domain boundaries opposite to the no-slip walls. With these conditions, scalar variables (density ρ, phase ϕ) at the boundary nodes take the value of the adjacent fluid nodes. The same is done for the tangential components of vector variables (velocity $\vec{u}$), while their normal component is inverted.

In a LB simulation, this boundary condition can be implemented by assigning the populations of the boundary node using:

$$f_i = f_{\bar{i}}^F$$

where F indicates the nearest neighbor fluid node while $\bar{i}$ is a distribution *complementary* to i, defined through the following relation of the corresponding discrete velocities:

$$e_{\bar{i}\alpha} = e_{i\alpha}\left(1 - 2\delta_{\alpha\beta}\right)$$

where β indicates the plane of symmetry (x, y, z), and $\delta_{\alpha\beta}$ is the Kronecker delta.

The effective location of the plane of symmetry with a boundary condition implemented in this manner is mid-way between the boundary node and the nearest neighbor fluid node.

To start a simulation targeting a particular Ca, a procedure similar to the 2D case was used, with the 2D Poiseuille solution replaced by the streamwise velocity profile in a rectangular channel Eq. (3.4).

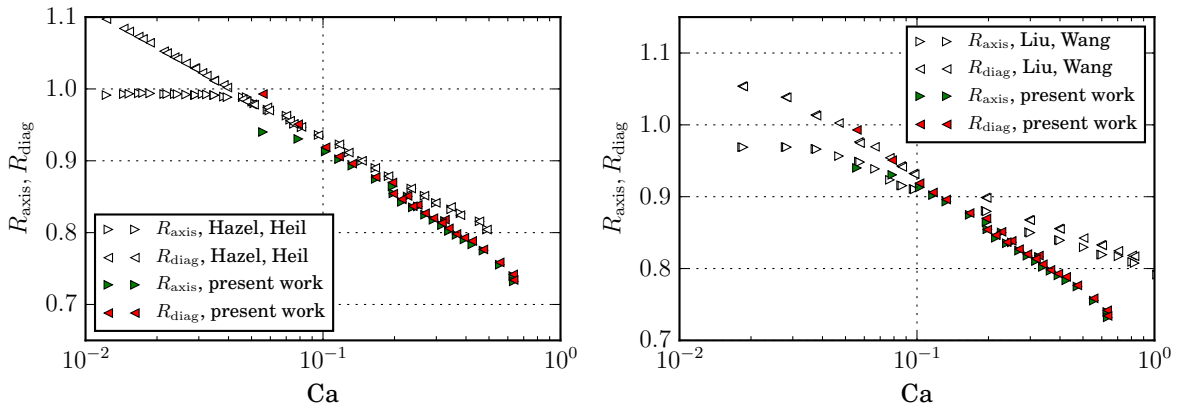


Figure 4.6: Comparison of the axial and diagonal bubble radii as a function of the capillary number Ca between data from the LB simulations and the results of Hazel and Heil [48] (left panel) and Liu and Wang [92] (right panel).

4.6.2 Bubble shape

Figure 4.6 shows a comparison of the results of our simulations with data from Hazel and Heil [48] and Liu and Wang [92] (data points were extracted from figures in these two papers using the Plot Digitizer software). We note that the LB simulations reproduce the bubble shape transition from an asymmetric to an axisymmetric shape with increasing Ca. The critical value of the capillary number at which this transition takes place can be estimated as $\text{Ca}_{\text{crit}} \approx 0.09$. This is in agreement with the findings of Liu and Wang [92], but different from $\text{Ca}_{\text{crit}} = 0.04$ reported by Hazel and Heil [48]. We note however that for $\text{Ca} > 2 \cdot 10^{-2}$ the bubble radii values from our simulations are closer to those of Hazel and Heil [48].

Figure 4.7 compares the radii along the bubble length for 3 chosen capillary numbers between our simulation results and that of Liu and Wang [92]. Note that the comparison can only be qualitative as the available Ca do not match exactly. Nevertheless, the overall shapes can be seen to exhibit the same trends. For instance, a small peak exists in the diagonal radius close to the rear end of the bubble for low capillary numbers.

4.6.3 Relative velocity

Relative velocity w is an important quantity in practical mass flow characterization [81, 159] and is defined as

$$w = \frac{u_{\text{bubble}} - u_{\text{ls}}}{u_{\text{bubble}}} \qquad u_{\text{ls}} = \frac{\int_A u_{\text{liq}} dA}{A} \tag{4.2}$$

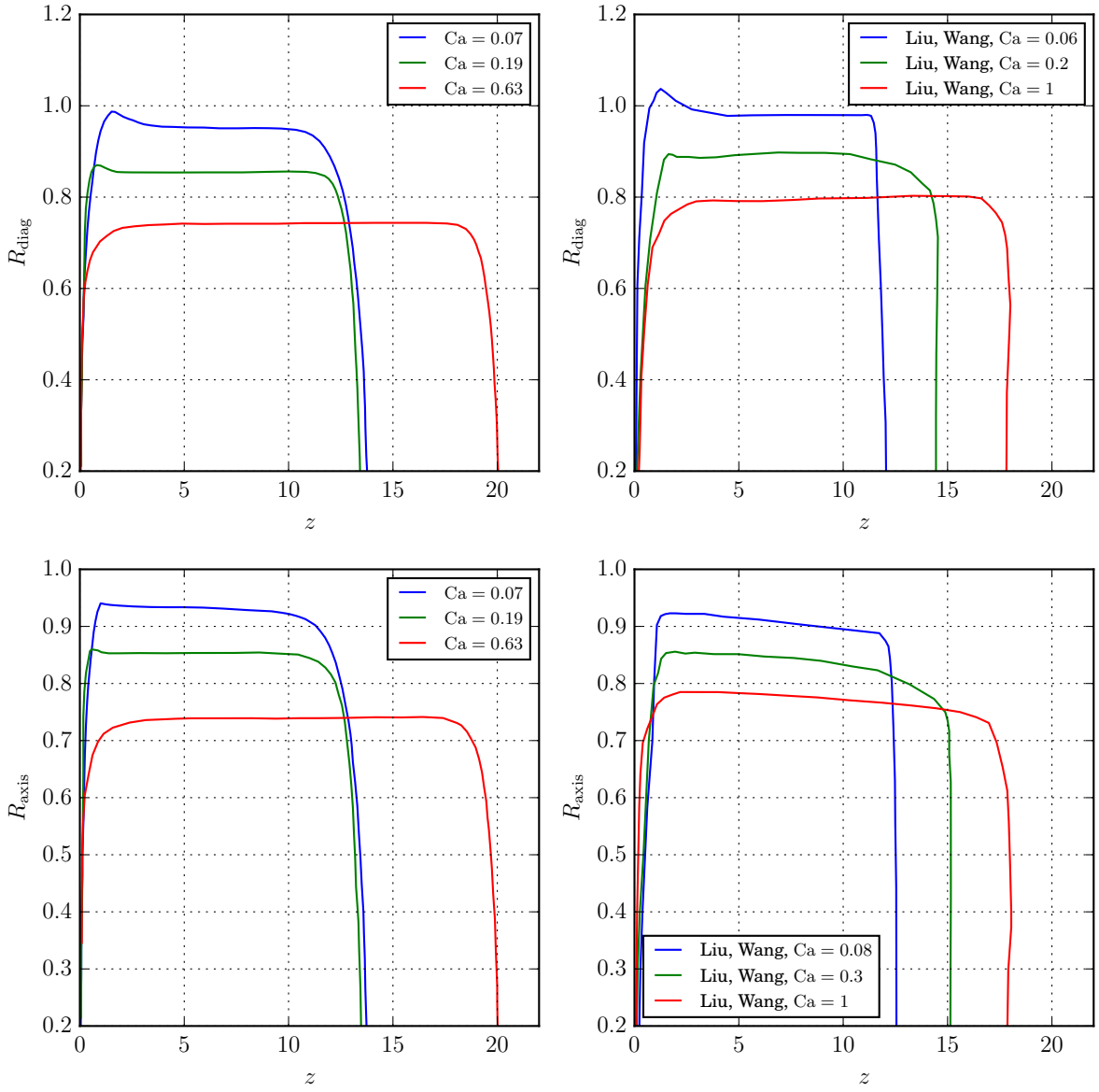


Figure 4.7: Variation of the bubble radii along the bubble length for different capillary numbers. Data from [92] (right column) was extracted using the Plot Digitizer software.

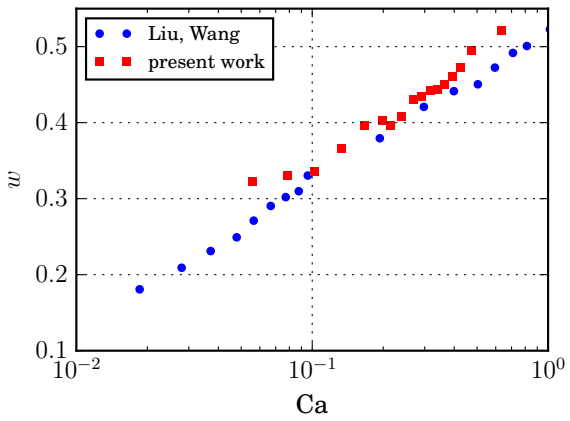


Figure 4.8: Comparison of the relative velocity w between present simulations and the results of Liu and Wang [92].

where u_{ls} is the liquid superficial velocity and u_{bubble} is the bubble interface velocity. As the bubble always moves faster than the surrounding liquid, $w > 0$.

We used the midpoint in the streamwise direction between two neighboring bubbles to compute u_{ls}, and compared w with the results of Liu and Wang [92] [92], obtaining good agreement (see Figure 4.8).

4.6.4 Rectangular channels

In addition to the main simulations of channels with a square cross section, we also performed simulations for rectangular channels with aspect ratios $\alpha = a_x/H \in \{1.2, 1.4, 1.6, 1.8\}$, where a_x is the channel size in the x direction and H is the channel height.

Figure 4.9(a) shows the dependence of the bubble radii in the x and y direction on the capillary number Ca and the aspect ratio α. The radii can be seen to increase with increasing α, confirming results of prior studies [48]. The radii also do not coincide ($R_x \neq R_y$) — for that to happen, the bubble length needs to be significantly longer than the one used here.

Hazel and Heil [48] studied the problem of Taylor flow in rectangular channels before and showed that regardless of the aspect ratio α, a simple 1:1 relationship exists between the capillary number Ca and a non-dimensional radius s_∞ defined as

$$s_\infty = \frac{2\sqrt{Q/\pi}}{H}\alpha^{-1/2} \tag{4.3}$$

where Q in the bubble area in the cross section. The physical interpretation is that if the gas is injected with the same velocity to channels with different aspect ratios, but

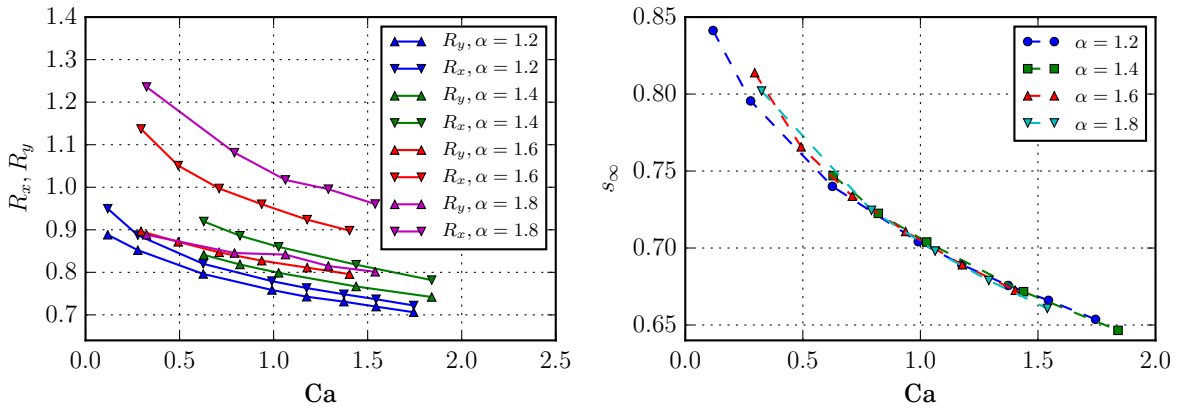


Figure 4.9: (Left panel) Bubble radii in the x and y directions for rectangular channels of different aspect ratios α. Radii are normalized by H. (Right panel) Non-dimensional radius (see Eq. (4.3)) as a function of the capillary number for simulations of bubble flow through channels with a rectangular cross-section of varying aspect ratios.

same cross-sectional area, the area occupied by the gas bubble should not depend on the aspect ratio. Figure 4.9(b) shows that the relationship holds with good accuracy for bubble train simulations in the free energy LB framework.

5 Hemodynamics

Hemodynamics is the study of blood flow. As of 2012, cardiovascular conditions are the leading cause of death in developed countries [59]. There is an ever-increasing amount of evidence linking the initiation and development of these diseases with the hemodynamical characteristics in the vicinity of the pathological changes in blood vessels [33, 34, 78, 93]. The limited resolution of medical imaging technologies, lack of direct access to physical quantities of interest (e.g. Wall Shear Stress (WSS)), as well as the need for disease risk assessment and impact prediction for medical interventions motivate computational approaches to the problem [14, 83, 135, 146].

The first decade of the 21st century is the first time when, due to continuous advancement of computing technologies, relatively fast and inexpensive, yet biologically accurate, patient-specific blood flow simulations become a viable option, bringing the promise of new possibilities for disease treatment and prevention [136].

We start this chapter with a short overview of the human cardiovascular system, which will provide the necessary background for further discussion. We then proceed to an overview of prior work in this field using the LBM, and discuss our results of hemodynamical simulations in one artificially generated geometry, and two geometries obtained from CT scans of patients from the Aneurisk database. We compare our results to ones obtained using a more traditional CFD method (FVM), showing overall good agreement and the potential of significant speed-ups when LBM and GPUs are used.

5.1 Biophysical background

5.1.1 Physical properties of blood

In animals, blood is responsible for the transport of nutrients and oxygen to cells, as well as metabolites away from them. Blood is a complex fluid formed by a suspension of cells in *blood plasma* — a solution of water (91%), ions (1%), proteins (7%), and other molecules. In healthy humans, plasma constitutes 54.3% of total volume, while Red Blood Cells (RBCs) and white blood cells take up 45% and 0.7%, respectively.

The complex composition of blood causes it to behave as a non-Newtonian, shear-thinning fluid. This can be considered as an evolutionary adaptation for reducing the resistance of flow through capillary vessels. The shear-thinning behavior of blood is apparent at low shear rates — above $100\,\mathrm{s}^{-1}$ the behavior becomes Newtonian [10]. At

shear rates very close to zero, blood can also be considered a Bingham plastic [17, 39].

Blood viscosity is determined primarily by hematocrit (volume percentage of RBCs in blood), RBC deformability, RBC aggregation, and plasma viscosity. Hematocrit is considered pathological when it is below 40% or above 50%.

Typical kinematic blood viscosity at normal human body temperature (37° C) is within the range $(2.8, 3.8) \cdot 10^{-6}$ m^2/s. Blood density, at $1060\,\mathrm{kg/m^3}$, is close to that of water, with the slightly increased density attributable mainly to RBCs.

Most blood flow is laminar at $\mathrm{Re} \leq 300$, but turbulence is possible with high flow rates in the descending aorta (in athletes) and in some pathological conditions (heart murmurs, stenotic heart valves with plaque buildup on the valve).

5.1.2 Red Blood Cells

RBCs are arguably the most important biological component of blood, mainly due to their prevalence (the next most common biological component, the leukocyte is approximately 500x less numerous), overall impact on blood viscosity, and the core function of oxygen transport (accomplished via hemoglobin).

RBC are metabolically active cells, but they contain no nucleus, mitochondria or RNA. They do not reproduce themselves, but are produced from stem cells in bone marrow. A human RBC lives for approximately 125 days, after which it is destroyed by apoptosis. RBCs can also be destroyed earlier in the process of *hemolysis* — e.g. biologically in case of infections or autoimmune disorders, or mechanically, in case of surgery or in medical implants. In healthy humans there are about 5 million RBCs in every mm^3 of blood.

From the physical point of view, RBCs are biconcave disks, $2\,\mu$m thick, $8\,\mu$m in diameter, and $85 - 90\,\mathrm{mm}^3$ in volume. The cell membrane is highly deformable (RBCs can pass through capillaries with an inner diameter of $4\,\mu$m) and can be modeled as neo-Hookean [56]. The cytoplasm (interior) of RBCs is a fluid of viscosity of $6 \cdot 10^{-3}\,\mathrm{Pa \cdot s}$, about 5 times higher than that of the plasma in which the RBCs are suspended. The RBC shape is considered to be evolutionarily optimized for large surface area, as the cell surface is where the hemoglobin molecules are located.

When modeling blood flow, it is typically assumed that flow through vessels larger than $200\,\mu$m in diameter can be modeled using homogeneous fluid. For fluid flowing through arterioles and venules smaller than $25\,\mu$m, as well as through capillaries, RBCs have to modeled as discrete objects [113].

5.1.3 The cardiovascular system

The Cardiovascular System (CVS) is a bodily system whose functions are to deliver nutrients and oxygen to all cells in the body, remove cellular waste and regulate the body temperature. The CVS can be subdivided into 3 subsystems:

- pulmonary circulation – blood flow through the lungs, supplied by the pulmonary artery,
- systemic circulation – blood flow through the rest of the body, supplied by the aorta,
- coronary circulation – a specialized system for the heart.

It is notable that the coronary circulation subsystem, while accounting for less than 5% of cardiac output, is a frequent cause of disease. The conditions in this subsystem can also dynamically change, with blood flow increasing by a factor of 4-7x during exercise in young people.

The CVS is powered by the heart, a central muscle driving blood flow throughout the body in a cyclic pattern. The cardiac cycle is divided into two general phases, characterized by different blood pressures — *diastole*, when the heart chambers are being refilled, and *systole*, when blood is ejected from the heart.

Blood vessels

Unless injury occurs, blood circulation takes place within blood vessels. Arteries (through which oxygenated blood flows) have 3 layers: *intima* (single layer of endothelial cells and a supporting layer of internal elastic lamina), *media* (composed of smooth muscle cells, which can actively expand or contract, as well as passive elastic tissue), and *externa* (composed of passive connective elastic tissue) [125]. A typical artery wall thickness is 10-15% of the lumen radius. Arteries can be subdivided into 3 groups: elastic (can expand and act as a capacitance vessel), muscular (enable vasodilation and vasoconstriction to regulate blood pressure), and arterioles (smallest arteries with a diameter of less than 0.5 mm).

Arteries are non-Hookean, becoming stiffer with stress, and viscoelastic, with stress depending on load, area and rate of strain. Their stiffness increases with age, approximately by a factor of 3 over a human lifetime. Large arteries experience pulsatile flow and therefore are subject to stretching under hemodynamic load. For instance, the thoracic aorta is subject to largest circumferential stretch of up to 30% [144], and the base perimeter of the aortic valve varies by approximately 20% at normal blood pressure. In smaller-diameter arteries and arterioles, the pulsatile components of blood flow are small, going to zero in capillaries where the flow can be considered steady.

Capillaries differ from large blood vessels in being composed only of an $0.5\,\mu$m-thick endothelial layer and a basement membrane, and having a very small lumen diameter of just $5 - 9\,\mu$m. The human body contains about 10^{10} capillaries, with an average length of 1 mm. Together, they hold around 500 ml of blood. Most body cells are within $20\,\mu$m of a capillary. Intercellular clefts between endothelial cells make it possible for water and water-soluble ions to diffuse to the surrounding tissue. The cleft size is on the order of several nanometers and varies depending on location inside the body (e.g. smaller in the brain, larger in the liver).

Common medical conditions

A complex system like the CVS can be subject to many failure modes and disturbances, which can lead to serious illness. Some of the most common medical conditions of the CVS include:

- atherosclerosis – narrowing and occlusion of blood vessels, affects mainly medium or large arteries,
- aneurysms – abnormal bulging of blood vessel walls, caused by weakness/thinning of the wall; typically occurs in the abdomen in the lower part of the aorta, or in the cerebral vasculature,
- stenoses – a general concept referring to obstruction of flow.

Atherosclerosis is characterized by an increase in stiffness and decrease in compliance of blood vessel walls. Untreated, it can lead to heart attack or stroke. Currently available treatments include stents and bypass grafts. The idea of the latter is to reroute blood flow around an obstructed vessel through a new surgically implanted vessel. Bypass grafts are typically applied in case of coronary artery diseases. Stents are small mechanical devices inserted into arteries to provide structural support, make arteries wider, and increase blood flow through the vessel.

The most widely accepted hypothesis is that atherosclerosis starts with endothelial injury or dysfunction. This leads to high cell turnover, and creation of so called leaky junctions, which enhance the permeability of the endothelium to macromolecules (such as LDL) and some immune system cells (monocytes).

In other diseases of blood vessels, such as thoracic aortic aneurysms, noninflammatory factors are suspected to be at play [61]. Specifically, genetic mutations causing mechanosensing defects, which in turn lead to maladaptive remodeling of the vessel wall through pathways that reduce its load-bearing capability and can lead to eventual rupture.

Studies have shown that the onset of atherosclerosis can be directly linked to prolonged exposure to low WSS [28]. This typically happens at outer walls of vessel bifurcations,

and areas of flow recirculation. A typical mean value of WSS in large arteries in humans is 1.5 Pa [123] which is 5 orders of magnitude lower that the circumferential stress experienced by the artery.

The biological explanation of the link between WSS and wall pathologies is based on the idea of arteries trying to maintain mechanical homeostasis. This process is mediated by endothelial cells, vascular smooth muscle cells and fibroblasts, which sense mechanical signals such as the wall stress and control wall remodeling through chemical pathways and altered gene expression [29, 119]. Within the framework of this theory, arteries are expected to regulate their lumen to maintain an approximately constant WSS close to a preferred value. When the WSS is increased, the artery reacts by increasing smooth muscle proliferation and synthesis of collagen, which reinforce the vessel wall, allowing for thickness increases and expansion of the lumen. WSS reductions on the other hand are linked will smooth muscle cell death, thinning of the wall, and lumen decrease [60]. The initial vasoactive response to altered WSS takes place over a period of several days. If the new flow conditions are sustained, remodeling of extracellular matrix and smooth muscle cells follows after approximately two weeks. Under constant flow conditions, mature healthy arteries exhibit little cell turnover ($< 1\%/$ day), and extracellular matrix components have a half-life of years to decades.

5.2 Lattice Boltzmann hemodynamical simulations

In what follows, we compare the results of fluid flow simulations in an artificial geometry and two realistic geometries from the Aneurisk database [1] obtained with LBM and FVM.

All FVM simulation results presented here are courtesy of Jakub Poła. The simulations were conducted using the open source C++ OpenFOAM suite [111], GPU-accelerated with the commercial SpeedIT Flow package [141, 147], obtained thanks to the kindness of the Vratis company. Time-dependent simulations used the Pressure-Implicit Split-Operator (PISO) algorithm [63] with an adjustable time step size, while stationary simulations used the Semi-Implicit Method for Pressure Linked Equations (SIMPLE) algorithm [114]. LBM simulations were performed using the Sailfish package.

The primary motivation for our work was to validate our LBM solver in the context of hemodynamics against a more established CFD method (FVM), as well as to obtain performance estimates indicating which method can provide results more quickly when GPU are used. As discussed before, simulation speed is a crucial factor on which clinical applicability of these methods will depend.

5.2.1 Prior work

Hemodynamics has been an application area of the LBM since shortly after the method was first introduced. Krafczyk et al. [79] performed simulations of blood flow though a model of an artificial aortic valve. The leaflets of the valve were assumed to have a fixed position. The data was not cross-checked with any other studies, but the overall conclusion was that the resulting profiles looked good.

Other studies included flows in idealized model geometries [3, 5, 7, 40, 155], simple 2D models of stents [55], as well as realistic geometries generated based on medical images (computed tomography angiography, magnetic resonance angiography) of patients, mainly focusing on the human abdominal aorta [4, 6, 8]. Xiu-Ying et al. [155] compared various 3D lattices, concluding that D3Q19 provides a good trade-off between memory use and precision. More recent works [40, 160] also analyzed the impact of advanced relaxation models such as ELBM, MRT and RLB. Geerdink and Hoekstra [40] found that with MRT it was hard to predict what combination of domain size and flow speed would result in a stable simulation, concluding that LBGK performs best for most arterial flows, unless high Re flow is present, in which case they recommended ELBM instead. Závodszky and Paál [160] performed simulations of flow through a model of the Internal Carotid Artery (ICA), finding RLB and incompressible LBGK to be unstable. MRT and ELBM were stable, but ELBM was noted to underestimate the velocities in several regions of the fluid domain.

All the works discussed above were concerned with relatively small scale simulations of very specific parts of the CVS. LBM has however also been applied in large-scale simulations of the entire left coronary artery tree [104, 116]. The geometry was generated from computed tomography angiography, acquired with a resolution of 0.5 mm, which was also taken as the diameter of the smallest vessels considered in the simulation. The simulation covered multiple physical scales, from 5 cm down to 10 μm and used 10^9 active fluid nodes in a bounding box of $3 \cdot 10^{11}$ nodes. Vessel walls were implemented with half-way bounce-back walls, while the inlet and outlets used Zou-He velocity and pressure boundary conditions, respectively. Some simulation runs included RBCs modeled as rigid ellipsoids limited to six degrees of freedom, with up to $3 \cdot 10^8$ objects within the simulation domain. Peters et al. [116] ran their simulation on a IBM Blue Gene/P computer, reaching a performance of 2 GLUPS to 125 GLUPS, depending on the number of cores used (4096, and 294 912, respectively). They noted that they ran their simulations for only 200 time steps, which is surprising, as for a lattice of this size, many more time steps would be necessary to reach convergence. Melchionna et al. [104] used a cluster with 1000 GPUs, reaching convergence within 20 minutes to 3 hours, depending on the features active in the simulations. The authors also noted that in their experiments a grid resolution of 20 μm was necessary for the WSS results to converge.

Relatively little work has been done on direct comparisons between LBM and other

methods. He et al. [52] compared flow in the middle cerebral artery (MCA) between an LBGK solution and an FVM solution obtained with OpenFOAM. Prior to the comparison, they validated their setup on the bend duct problem. Their MCA simulation used different inlet boundary conditions between the two methods — a plug profile in FVM and parabolic profile with matched overall flux in LBM. This was done because the LBM simulation was unstable when the plug profile was used. The inflow velocity was time-dependent and used a realistic waveform profile. The results between LBGK and FVM were in good agreement with deviations up to 7% in pressure and 2.5% in velocity in single probe points. No information was provided on spatially extended profiles, but a comparison of WSS was presented, showing overall similar structure. The simulation used $3.25 \cdot 10^5$ tetrahedral mesh elements in FVM and $4.33 \cdot 10^5$ cells in LBM, and took $\sim$ 1.2 days to simulate a single cardiac cycle for both methods (with FVM being marginally slower) [52].

Axner et al. [9] simulated blood flow in the abdominal aorta and the superior mesenteric artery, and compared LBM results with those obtained with the FLOTRAN package and the Finite Element Method (FEM). The geometry was generated based on on magnetic resonance images, and used 10^5 nodes for FEM and $7.4 \cdot 10^4$ fluid nodes for LBM. Walls were implemented with the bounce-back condition and the physical parameters of the flow were $\mathrm{Re}_{\max} = 3300$ and $\alpha \approx 11$. A time-harmonic flow was simulated and profiles on three different transverse planes were compared between the two methods, showing differences of less than 10%. The authors did not list any specific performance data, but noted that computation times between the two methods were comparable, and that LBM required less memory.

Závodszky and Paál [160] compared experimental particle image velocimetry and laser Doppler anemometry measurements in a model of the ICA to simulation results of a Finite Volume Method (FVM) simulation performed with Ansys CFX and MRT simulations with Sailfish and Palabos. Their FVM mesh used $8 \cdot 10^5$ cells, while the LBM lattices had between $1.57 \cdot 10^6$ and $8.17 \cdot 10^6$ active nodes. The conclusion of the study was that the LBM simulations were better at reproducing the experimental measurements than FVM.

5.2.2 Boundary conditions

One crucial factor that can be expected to impact the accuracy of the results of any hemodynamical simulation is the implementation of no-slip boundary conditions used to represent the vessel. The basic choice in this context is between a simple bounce-back condition, or a more advanced condition, which usually uses some form of interpolation to enhance accuracy and model the wall position at a subgrid level.

Bounce-back methods are known to be of second order accuracy when the wall is aligned with the lattice axes, but they degrade to first order in the general case which is of

relevance in hemodynamical simulations. The use of a staircase approximation also introduces a first order modeling error. It should however be noted that for hemodynamic flows in realistic geometries, the geometry itself is typically obtained from voxel-based imaging methods such as computed tomography (CT) with a spatial resolution significantly lower than that used in a LBM simulation. As such, the precise location of the wall is not exactly known and some modeling error is introduced already in the process of preparing the geometry for the simulation.

Carver et al. [21] studied the accuracy of various methods of imposing the no-slip condition in non-axis aligned stationary (Poiseuille) and pulsatile (Womersley) cylinder flows. The analyzed boundary conditions included the simple bounce-back method (BB), the Bouzidi-Firdaouss-Lallemand method (BFL), the Guo, Zheng, and Shi method (GZS), and the Junk and Yang method (JY). Flows were analyzed at $\mathrm{Re} \in (1, 100)$ and $\alpha \in (4, 12)$. The authors found out that the GZS and JY methods destabilize the simulation already at $\mathrm{Re} = 30$ and such are not applicable for realistic hemodynamical flows. BB and BFL had similar convergence properties, but BFL consistently achieved a 30% lower error than BB.

Artoli et al. [3] similarly showed showed that BFL enhanced the accuracy of velocity profiles in simple artificial geometries. However, they noted that BFL did not improve the accuracy of the fluid stress tensor in comparison to BB. BFL was also criticized for its lack of mass conservation and worse numerical stability [40]. Opposite conclusions were also reported, with BFL claimed to increase accuracy and convergence of WSS calculations [67], and to enhance stability and reduce simulation time [7]. Despite a potentially higher accuracy of the BFL condition, some authors note that BFL is too computationally costly if moving boundaries are used [7, 133] and that BB provides reasonable accuracy for practical applications and is much easier to use with real geometries [7, 40, 67, 133].

Given these conclusions, we decided to use the BB boundary condition for all simulations presented later in this chapter.

5.2.3 Wall shear stress

In a LB simulation, the shear stress tensor σ can be computed at every node using local distributions f_i:

$$\sigma_{\alpha\beta} = -p\delta_{\alpha\beta} - \left(1 - \frac{1}{2\tau}\right) \sum_i (f_i - f_i^{\mathrm{eq}}) e_{i\alpha} e_{i\beta}. \tag{5.1}$$

This can be considered as an advantage of LB over traditional NSE solvers, where the calculation of the tensor components requires the evaluation of velocity derivatives, which is a non-local operation.

The physiologically relevant WSS is a vector $\vec{\tau}_w$, defined as:

$$\vec{\tau}_w = \mu \left. \frac{\partial \vec{u}}{\partial y} \right|_{y=0} \tag{5.2}$$

where y is the wall-normal coordinate and $\vec{u}$ is fluid velocity parallel to the wall. The wall shear stress is measured in units of pressure (Pa). The WSS can also be computed through a contraction of the stress tensor σ with the wall-normal vector $\vec{n}$

$$\tau_{w\alpha} = \sigma_{\alpha\beta} n_\beta - \sigma_{\gamma\beta} n_\beta n_\gamma n_\alpha. \tag{5.3}$$

It is easy to see that the first term of Eq. (5.1) does not contribute to $\tau_{w\alpha}$.

Depending on how the geometry of the simulation was obtained, $\vec{n}$ may be difficult to determine. In easy cases, the geometry of the wall can be given analytically, in which case $\vec{n}$ can be evaluated exactly. In realistic geometries obtained from experimental data, this is however not the case, and one is presented with three options:

1. compute $\vec{n}$ outside of the LB framework, and map it to the boundary nodes of the voxel LB geometry,
2. estimate $\vec{n}$ based on the LB geometry,
3. estimate $\vec{n}$ based on the fluid velocity field.

Option 1 above will provide the highest quality estimate, but it requires access to data outside of the LB simulation. For instance, experimental data from medical imaging could be preprocessed to generate a surface mesh, where the normal vectors are given for every face. If the imaging modality generates voxel data natively, this additional step might introduce unnecessary complexity into the simulation workflow. Similarly, if a dynamic geometry is desired (e.g. the wall location changes in the course of the simulation due to deposition/erosion processes, or due to vessel distensibility), an external mesh could not be readily available.

To deal with this issue, at least two approaches were proposed in the literature. The *dynamic* method [133] is based on the observation that $\vec{n} \cdot \vec{u} = 0$ holds at a non-slip wall. The *geometric* method [99] defines 6 discretized normal vectors that can be assigned to an internal facet of a wall boundary node within the LB lattice, and then averages these vectors within a neighborhood of a given radius r to compute the final $\vec{n}$.

Stahl et al. [133] analyzed their dynamic approach with flows in a bent pipe and a straight pipe inclined at various angles, but they did not test it with time-dependent flows. It is therefore not clear how well the dynamic method performs under flow conditions realistic for blood vessels. For highly disturbed flows, the obtained $\vec{n}$ estimate could be expected to vary in time, leading to spurious results if the geometry of the flow is actually static. The dynamic method is also sensitive to wall roughness caused by the staircase boundary approximation present when bounce-back conditions are used. The authors noted that

accuracy can be increased by using interpolated boundary conditions for the walls [18], as well as computing WSS up to 5 lattice nodes away from the wall.

Matyka et al. [99] showed that the geometric method leads to a relative normal vector error 50% lower than that obtained with the dynamic method. Interestingly, they also showed that in the inclined channel test, the WSS estimate did not change significantly between the two estimation methods and the use of the exact wall normal, indicating that errors in the flow field itself in the vicinity of the wall are the dominating source of inaccuracies. In a further test of flow though a realistic geometry representing the human aorta, they also showed good agreement between results obtained with the geometric method in Sailfish and an OpenFOAM FVM simulation.

The accuracy of WSS in LB simulations has also been studied in both axis-aligned and inclined channels, concluding that WSS is usually more than first order and less than second order accurate, depending on the boundary conditions used [67, 82]. Kang and Dun [67] similarly to Stahl et al. [133], noted that evaluating WSS at a distance of 2 nodes away from the wall yielded the smallest error. The use of interpolated boundary conditions [18] was also shown to increase the level of convergence. All works note the troubles associated with the use of these boundary conditions in realistic geometries though, and conclude that the use of simple bounce-back is sufficient in most cases.

A derivative quantity related to the WSS is the Oscillatory Shear Index (OSI), defined as:

$$\mathrm{OSI} = \frac{1}{2}\left(1 - \frac{\left|\int_0^T \vec{\tau}_w dt\right|}{\int_0^T |\vec{\tau}_w|\, dt}\right), \tag{5.4}$$

where T is the cardiac cycle length. High OSI, indicative of time-varying WSS is predictive of atherosclerotic plaque development [22].

5.3 Test cases

We now proceed to simulate fluid flow in various geometries, at conditions resembling those found in healthy humans. The principal goal of these simulations is to compare the performance and quality of the LBM solution to one obtained using a more established numerical method — the Finite Volume Method (FVM). For simplicity, and to keep factors impacting the effectiveness of the solution separate, we assume the vessels to be non-distensible, and the fluid to be Newtonian.

5.3.1 Simulation setup

Geometry

For all simulations, the geometries are described by a surface mesh stored in an STL file. For FVM simulations, the STL file served as a base for the generation of a volumetric mesh, the details of which were case-specific. For LBM, the surface mesh was mapped onto a dense grid of nodes using an octree-based voxelizer supplied in the 2012-04-22 release of the CVMLCPP library.

Boundary conditions

For pipe flows, the velocity profile is flat in the middle of the pipe in the entrance region ($L_e/D \approx 0.06\mathrm{Re}$), and then fully develops into a Poiseuille flow. The Poiseuille law applies as long as turbulence does not develop, which for pipe flows is $\mathrm{Re} \lessapprox 2000$. In the entrance region, the pressure gradient is spatially varying, unlike in a fully developed Poiseuille flow.

In our simulations we assume that the simulated system is just a part of a larger vessel network — i.e. we assume that the vessels continue beyond inlets/outlets outside of the simulation domain, without any significant changes to their geometry. Under this assumption, we use a velocity boundary condition with a parabolic velocity profile to drive the flows in both LBM and FVM simulations. The location of the apex of the parabola is computed as the center of a circle inscribed into the inlet surface. For transient flows, the inlet velocity time series is used to scale the maximum velocity of the parabolic profile.

The use of a parabolic velocity profile is also further motivated by prior studies [20] that explored flow simulations with parabolic, Womersley and blunt inlet profiles in patient-specific blood vessel geometries. These were compared to simulations using experimentally obtained patient-specific velocity profiles, and the overall conclusion was that the parabolic velocity profile produced results closest to the experimental data.

In LBM simulations, the equilibrium boundary conditions were used to set the inlet velocity, as well as to set a constant density of 1 on all outlet surface. No-slip walls were implemented using the standard half-way bounce back method. The initial state of the LBM simulation was set to the equilibrium distribution corresponding to a density field of 1 and a velocity field set to $\vec{0}$ everywhere except for the inlet nodes, where the velocity profile imposed by the boundary condition was used. It is clear that such a configuration is far from the target equilibrium. As a result, pressure waves are generated at the start of the simulation, and care has to be taken to ensure that they do not cause numerical instabilities. This issue is discussed in more detail in the analysis of the specific cases.

In FVM simulations, walls were implemented by using the zero-gradient pressure con-

dition and setting velocity to $\vec{0}$. The outlet used a constant pressure set to 0 and the zero-gradient condition for the velocity field. The zero-gradient condition for pressure was also used at the inlet.

5.3.2 Comparison with FVM

In order to be able to compare the performance of LBM and FVM, it is necessary to use convergence criteria that do not depend on the details of the used solver. Some commonly used criteria are method-specific — for instance, terminating the simulation once residuals reach a predefined level is a common method employed in FVM simulations, which does not have an LBM equivalent.

We begin with the relative L^1 and L^2 errors of the density and velocity fields for both types of simulations:

$$\epsilon_p = \frac{1}{N}\sum_{i=0}^{N}\left|\frac{p_i(t) - p_i(t-\delta)}{p_i(t-\delta)}\right| \qquad \epsilon_{\vec{u}} = \frac{1}{N}\sum_{i=0}^{N}\frac{|\vec{u_i}(t) - \vec{u_i}(t-\delta)|}{|\vec{u_i}(t-\delta)|}, \tag{5.5}$$

with the time delay $\delta = 1$ for FVM and $\delta = 500$ for LBM, and N being the number of volume elements in FVM and lattice nodes in LBM. These errors are well defined for both methods since their definition uses macroscopic fluid fields only, and these are available regardless of the used numerical method.

A possible stop criterion would be to require the errors in Eq. (5.5) to reach a specific level. However, it is easy to see that the definition still has a free parameter δ, for which no clear equivalence between LBM and FVM exists. The values we chose above were dictated by reasons of computational efficiency.

This problem can be avoided if the relative errors are defined with regard to a reference asymptotic solution instead of a time-delayed solution from δ iterations ago. To implement this idea, we first run the simulation long enough for ϵ_p and $\epsilon_{\vec{u}}$ to stop to change noticeably. This defines the asymptotic solution $p(\infty), \vec{u}(\infty)$ of a given solver. Graphically, it can be observed as a plateau in Figure 5.1. It is easy to see, that the asymptotic solution does not depend on the value of δ anymore as long as the plateau spans a time longer than δ.

The asymptotic solution can then be used to define a common termination criterion that can be used for both LBM and FVM:

$$\epsilon_p^{\infty} = \frac{1}{N}\sum_{i=0}^{N}\left|\frac{p_i(t) - p_i(\infty)}{p_i(\infty)}\right| \leq 10^{-4} \qquad \epsilon_{\vec{u}}^{\infty} = \frac{1}{N}\sum_{i=0}^{N}\frac{|\vec{u_i}(t) - \vec{u_i}(\infty)|}{|\vec{u_i}(\infty)|} \leq 10^{-4}. \tag{5.6}$$

We use Eq. (5.6) to measure performance of stationary simulations (see Figure 5.2 and compare against Figure 5.1 for an illustration how this works). For simulations where the

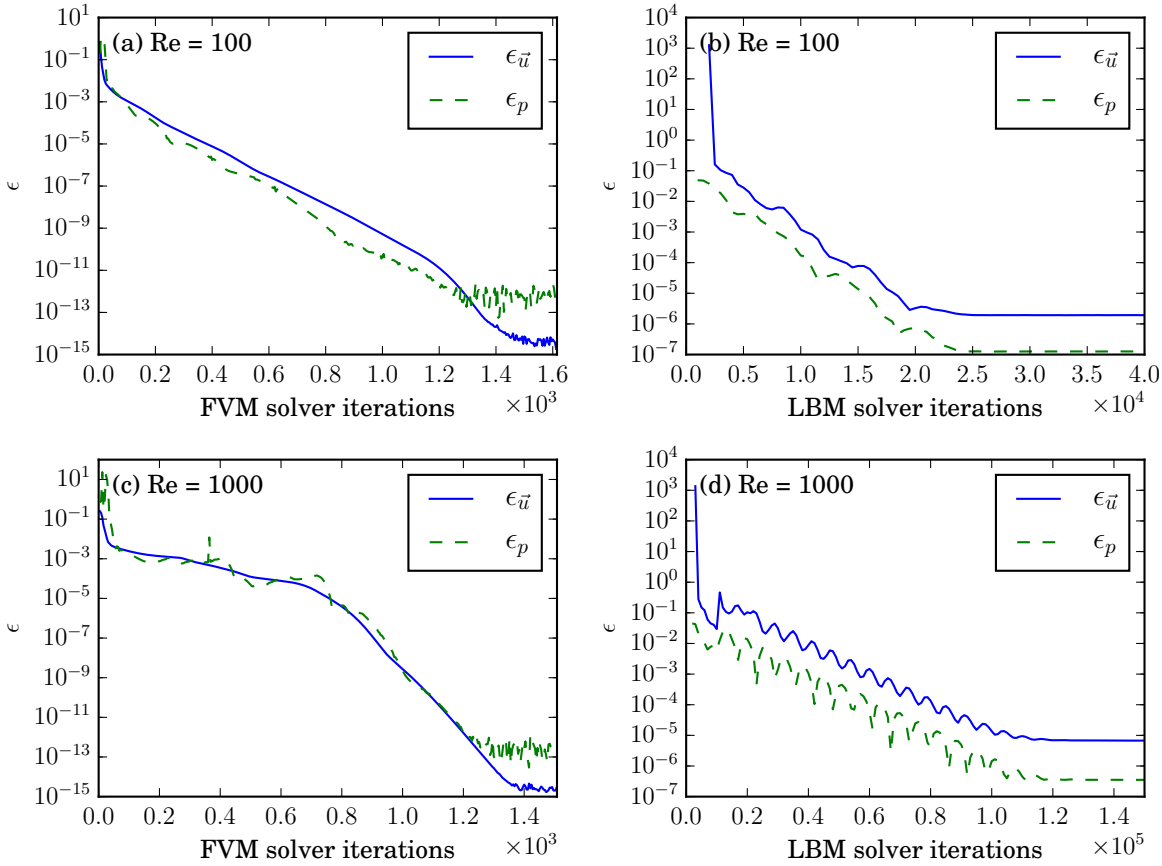


Figure 5.1: Error norms of velocity and pressure for the U-shape case at Re = 100 (panels (a) and (b) for FVM and LBM, respectively), and at Re = 1000 (panels (c) and (d)). In all cases, the simulation was terminated when both $\epsilon_{\vec{u}}$ and ϵ_p reached steady state. The oscillatory character of the error norm in the case of LBM is caused by a pressure wave propagating back and forth between the inlet and outlet, caused by the initial conditions.

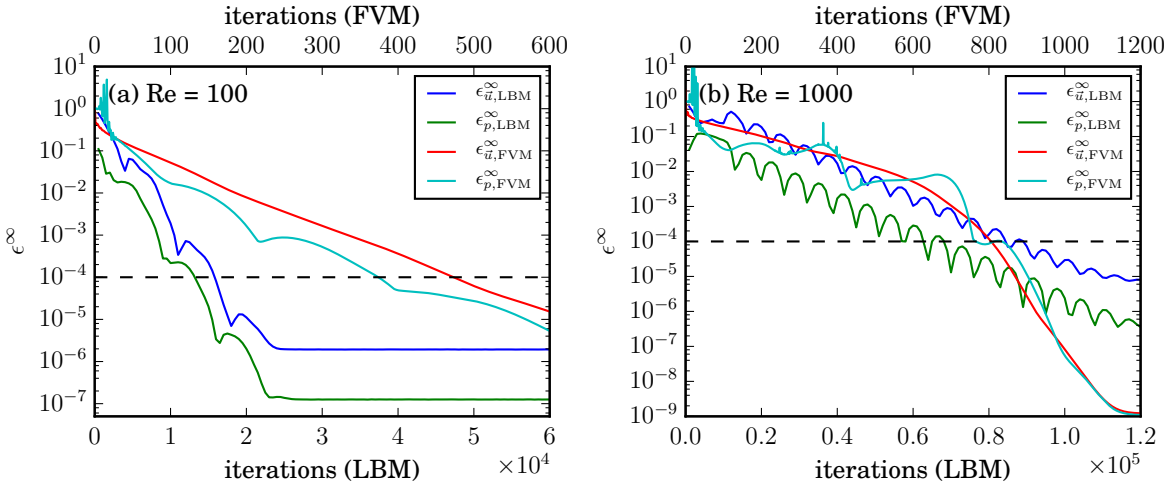


Figure 5.2: Error norms of velocity and pressure computed against the asymptotic solution for the U-shape case at Re = 100 (a) and Re = 1000 (b). The dashed horizontal line shows the new simulation termination criterion.

flow field is a function of time, we run the simulation until a full cycle is complete once initial transients stop being visible in velocity/pressure time series at slices of interest.

5.3.3 Artificial geometry

The first and simplest geometry we use is an artificially generated U-shaped pipe. The pipe cross section diameter is $D = 2.54$ cm. The curvature radius R_1 of the bow equals 7.62 cm and the length L of the arms is 25.4 cm (see Figure 5.3).

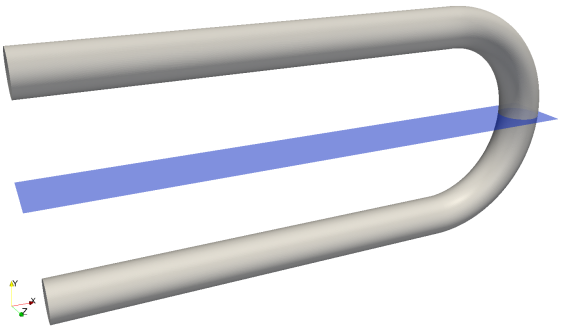


Figure 5.3: Artificial U-shape geometry. The blue plane indicates the symmetry plane from which measurements were taken for comparison with FVM simulations.

We simulate stationary flow at $\mathrm{Re} = 100, 1000$ and oscillatory flow at $\mathrm{Re}_{\mathrm{avg}} = 100$, with instantaneous Re oscillating between 90 and 110. In the transient flow simulation, the mean inlet velocity was set as:

$$\bar{u}(t) = u_0 \left(1 + 0.1 \sin(\omega t)\right), \tag{5.7}$$

with $u_0 = 0.0131\ \mathrm{m\,s^{-1}}$, and $\omega = 2\pi\ \mathrm{s^{-1}}$.

This simple geometry allows us to verify the correctness of our results, as well as to establish the methodology that will later be used to tackle more realistic shapes. This test geometry is also physiologically relevant as the bend of the pipe can be considered a prototypical model of the descending aorta.

The simulations used the incompressible LBGK model. At $\mathrm{Re} = 1000$, the initial pressure wave was strong enough to eventually destabilize the simulation close to the outlet, where it triggered a backward flow region on the outlet surface. To avoid this problem we started the $\mathrm{Re} = 1000$ simulation at a higher viscosity corresponding to $\mathrm{Re} = 100$, and lowered the viscosity to the target value after 10^4 iterations.

For FVM simulations, two volumetric meshes were used. The first one, with $2.57 \cdot 10^6$ hexahedral elements was used to simulate stationary flows. For transient flow such a dense mesh was not required, and a mesh of $5 \cdot 10^5$ elements was used instead.

Model selection

To determine the optimal parameters for the simulations we investigated the impact of adjusting the mean flow velocity, as well as using the incompressible LBGK model. The data from all runs was compared to ensure that no significant errors were introduced to the velocity and density fields because of these adjustments.

Figure 5.4 shows a summary of these experiments. In all cases, the use of the incompressible model significantly accelerates convergence. The same can be said about adjusting the mean flow velocity, which we were able to tune to the maximum value at which the simulation still remained stable. Overall, these optimizations cut the simulation time by almost a factor of 3 for $\mathrm{Re} = 1000$. For $\mathrm{Re} = 100$, the improvement is ≈ 2. It is interesting that no real improvement was seen beyond $u_0 = 0.11$ with the incompressible model, suggesting that $2.5 \cdot 10^4$ iterations is the minimum time necessary for the system to reach a steady state at the desired level of precision.

Spatial convergence

The quality of the solution of the LB simulation is determined by a number of factors, including the size of the lattice, the voxelization algorithm, boundary conditions used for the walls, inlets and outlets, as well as the Mach number of the flow. While the

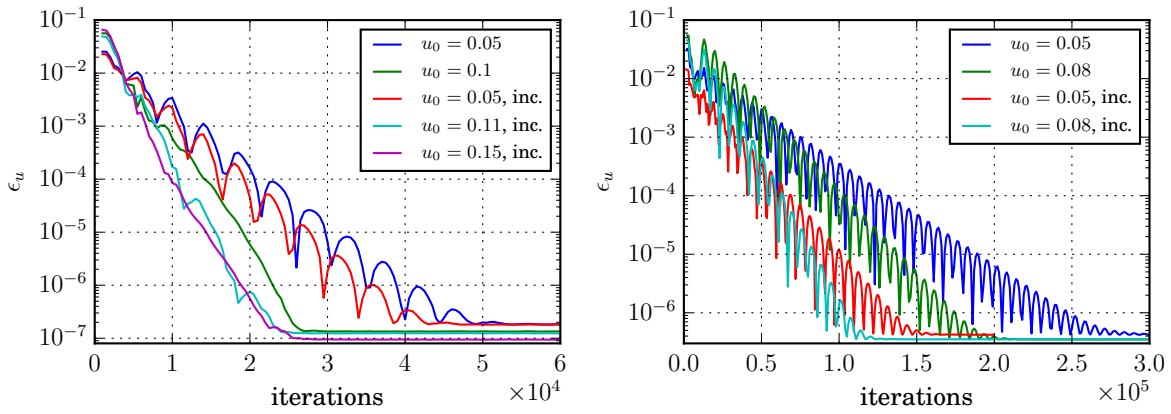


Figure 5.4: Impact of mean inflow velocity and relaxation model on simulation length until L^2 convergence is reached. Left panel: Re = 100. Right panel: Re = 1000.

impact of some of these factors is known when they are considered separately, there is no easy way of estimating their overall impact when taken *together* without running the actual simulation.

To determine the lattice size necessary to obtain a solution of satisfactory quality, we performed a number of simulations at increasing lattice sizes with a factor of 3.2 difference between the smallest and largest size (32× difference in the total number of nodes). The selection of the smallest viable lattice is important as it directly impacts the length of the simulation, which is proportional to the volume of the simulation domain.

To inspect simulation results, we concentrate on the symmetry plane of the geometry which intersects the bend perpendicularly to the flow direction (see Figure 5.3) and compare both 2D plots and 1D profiles of velocity and pressure. At Re = 100, Figure 5.5 shows that the structure of the flow is correctly resolved at all resolutions, and no significant details are lost even with the smallest lattice where the diameter of the pipe is just 38 nodes. At Re = 1000, the flow field becomes more complex (see Figure 5.6), and the lowest resolution is no longer enough to represent it with enough detail. We note though that the overall structure is still correct — i.e. velocity peaks are in the right locations, etc. The density field suffers from a high frequency oscillatory pattern caused by instabilities triggered at the walls. These patterns disappear when larger lattices are used, or when the relaxation model is changed to MRT. This happens because with larger lattices, a higher numerical viscosity is used, which then dampens out the initial pressure waves.

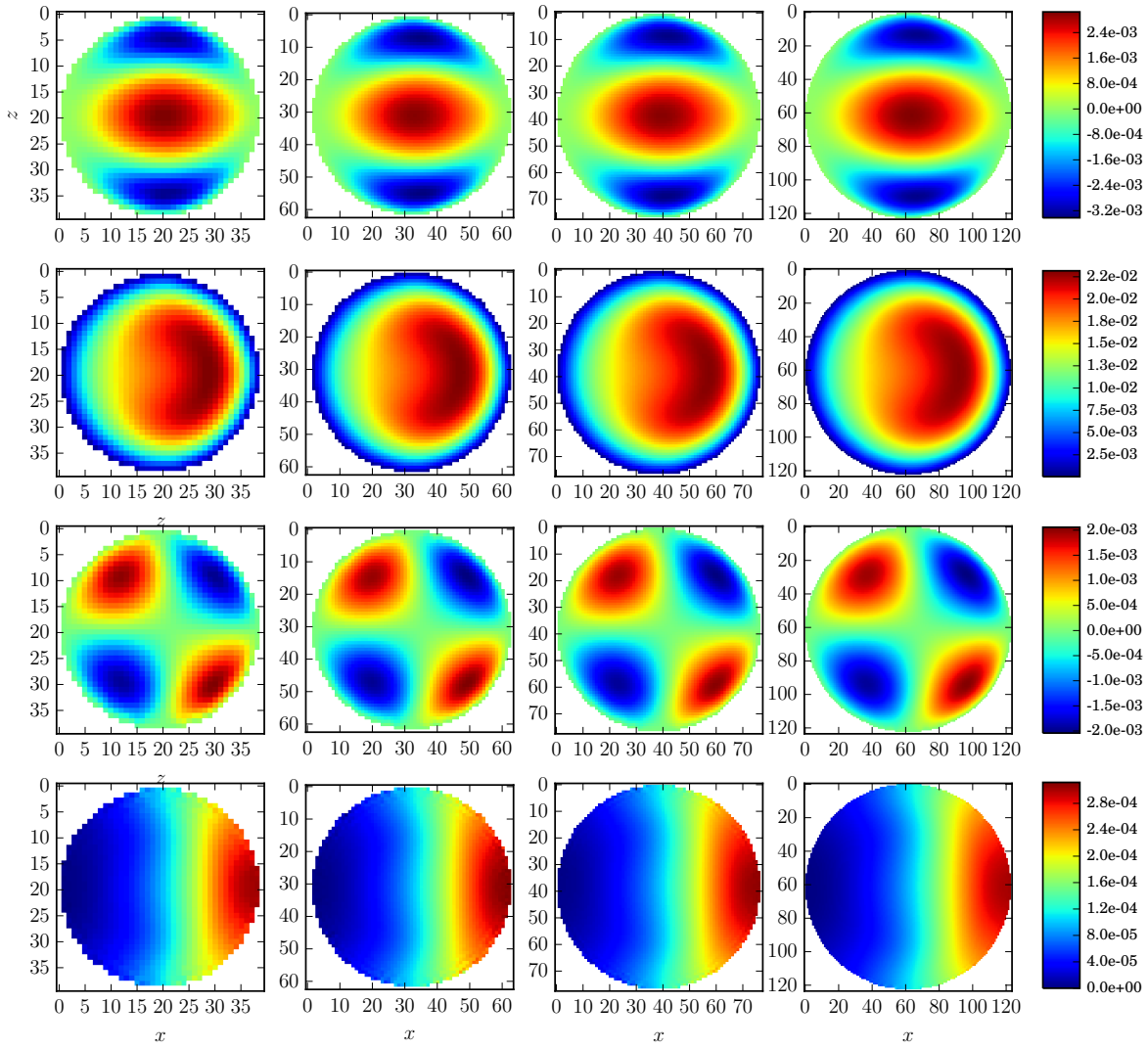


Figure 5.5: Velocity components (u_x, u_y, u_z in rows 1-3, respectively) and p (row 4) on the middle slice of the U-shape geometry at $\mathrm{Re} = 100$ and 4 different simulation resolutions: 38, 61, 76, 122 fluid nodes per pipe diameter in columns 1-4, respectively. All fields are shown in physical units — velocity in m/s and density-normalized pressure in $\mathrm{m}^2/\mathrm{s}^2$. Pressure is scaled so that 0 corresponds to the minimum value within the shown slice.

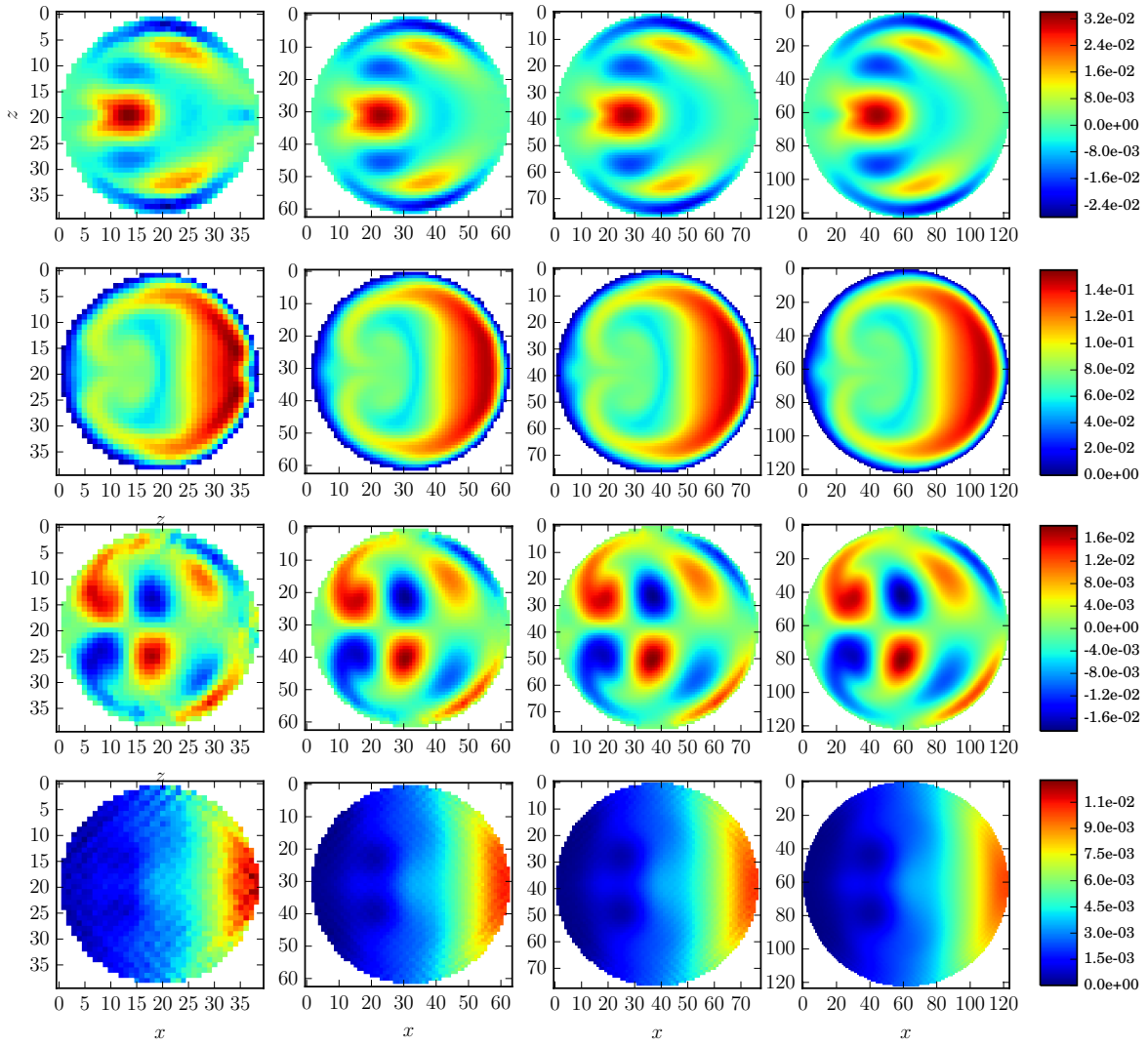


Figure 5.6: Velocity components (u_x, u_y, u_z in rows 1-3, respectively) and p (row 4) on the middle slice of the U-shape geometry at Re = 1000 and 4 different simulation resolutions: 38, 61, 76, 122 fluid nodes per pipe diameter in columns 1-4, respectively. All fields are shown in physical units — velocity in m/s and density-normalized pressure in m^2/s^2. Pressure is scaled so that 0 corresponds to the minimum value within the shown slice.

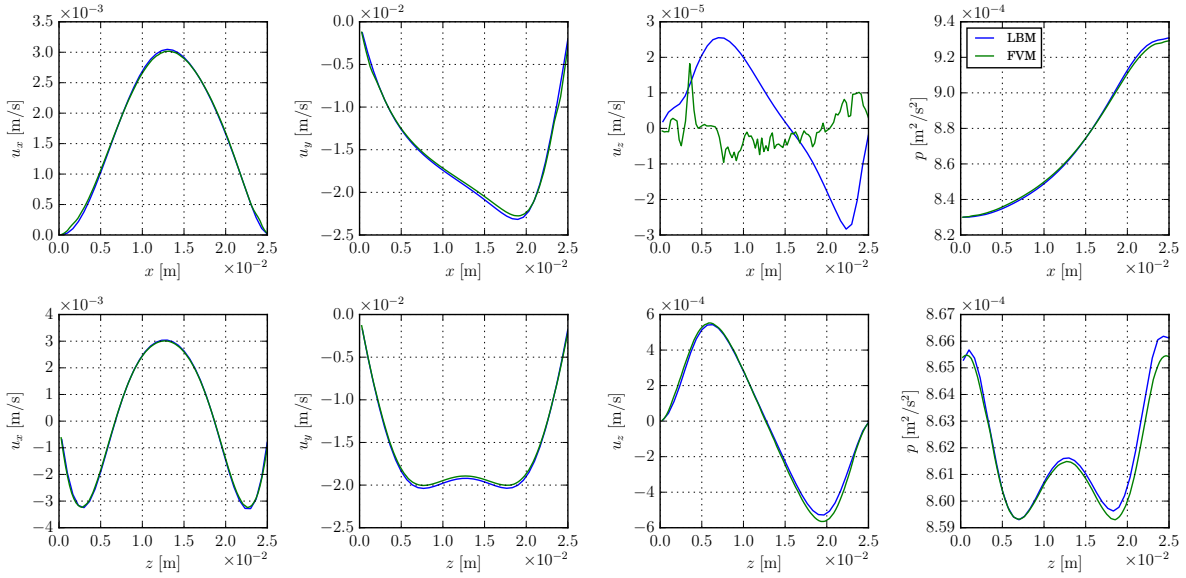


Figure 5.7: Profiles of velocity components and pressure on x and z slices through the middle slice of the U-shape domain, at Re = 100. Data from LBM and FVM is compared in physical units. Density-normalized pressure is shown.

Comparison with FVM

Stationary flow For a more qualitative evaluation of the results, we also compared the velocity and density profiles on the middle x and z sections of the previously discussed middle slice of the U-shape geometry. Figure 5.7 shows the results of Re = 100 simulations at 38 lattice nodes per pipe diameter, using the incompressible LBGK model at $u_0 = 0.11$. Good agreement in all profiles can be observed. The pattern visible in the u_z LBM results on the x-section is due to the first order interpolation used to compute the profile in the middle of the pipe as well as a slight asymmetry in the pipe cross-section introduced in the voxelization process.

The profiles at Re = 1000 show data from LBM runs at multiple lattice sizes, starting with the previously identified minimal resolution of 61 nodes per pipe diameter. While all velocity profiles are of acceptable quality, the density profile from the LBGK simulation suffers from the previously discussed oscillations. These can be eliminated by switching to the MRT relaxation model or increasing lattice size and viscosity. We also note that the high resolution LBM simulation (122 nodes per pipe diameter) confirms some of the cases where small discrepancies between LBM and FVM results are present (e.g. height of the u_x peak on the x slice). It also predicts some slightly (up to 10%) higher peaks and deeper troughs in the z slice velocity profiles. The high resolution LBM profiles are more symmetrical. Given the structure of the flow field in Figure 5.6 we believe them to be more precise than the FVM ones.

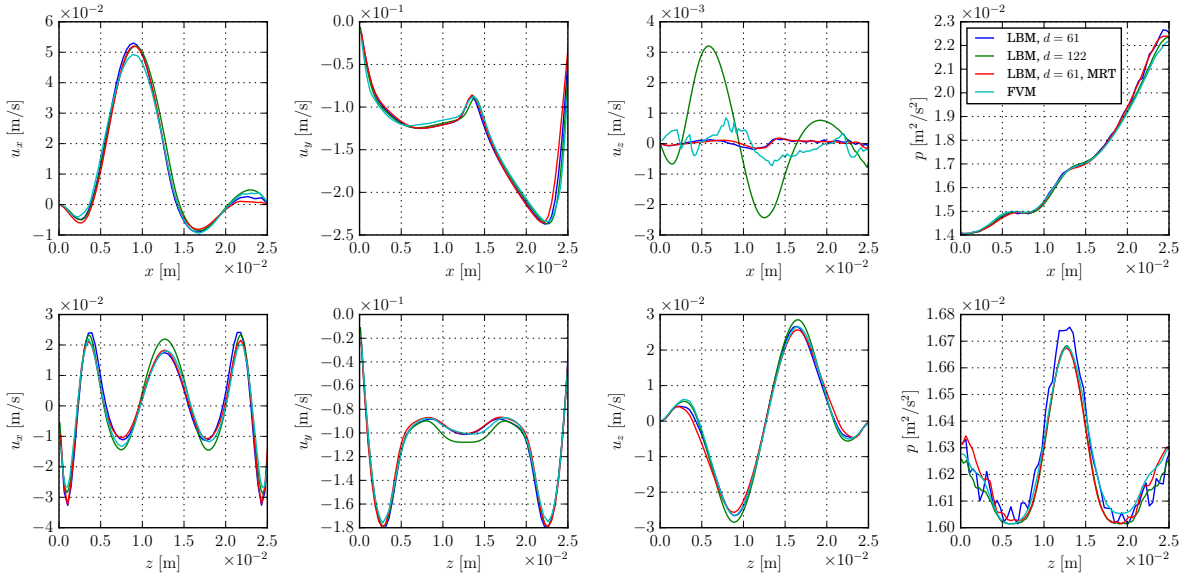


Figure 5.8: Profiles of velocity components and pressure on x and z slices through the middle slice of the U-shape domain, at Re = 1000. Data from LBM and FVM is compared in physical units. Density-normalized pressure is shown.

Oscillatory flow At the physical parameters of the oscillatory flow test case, $\mathrm{Re}_{\mathrm{avg}} = 100$ and $\alpha = 34.89$, corresponding to $f = 1$ Hz, a relatively low maximum velocity is necessary in order to keep the flow in a regime where compressibility artifacts do not affect the results. Using the smallest viable lattice size where the inlet diameter $d_{\mathrm{lb}} = 38$, our experiments showed that a $u_0 = 2.5 \cdot 10^{-4}$ was slow enough to yield good results. Due to the low overall fluid velocity, the density fluctuations seen in the simulation were of the order of the machine epsilon for single precision numbers, making it necessary to run the simulation with the round-off minimization model.

Since the low velocity u_0 at a fixed Re and domain size is linked to a similarly low LB viscosity, and since the latter controls the rate of decay of unwanted transients, we modified the initial conditions of the simulation. We set the initial values of the velocity and density fields to those from a previously completed stationary Re = 100 simulation with the same geometry. The fields were rescaled to match the new, lower value of the maximum velocity. The simulation was then run normally.

We then waited 8 cycles, or $6.29 \cdot 10^5$ iterations for the initial transients to disappear, and collected data for the remaining 2 cycles ($1.57 \cdot 10^5$ iterations). The results are presented in Figure 5.9 and can be seen to be in good agreement with FVM results. We also ran the simulation at lower u_0 and with larger lattices, getting matching results (not shown here).

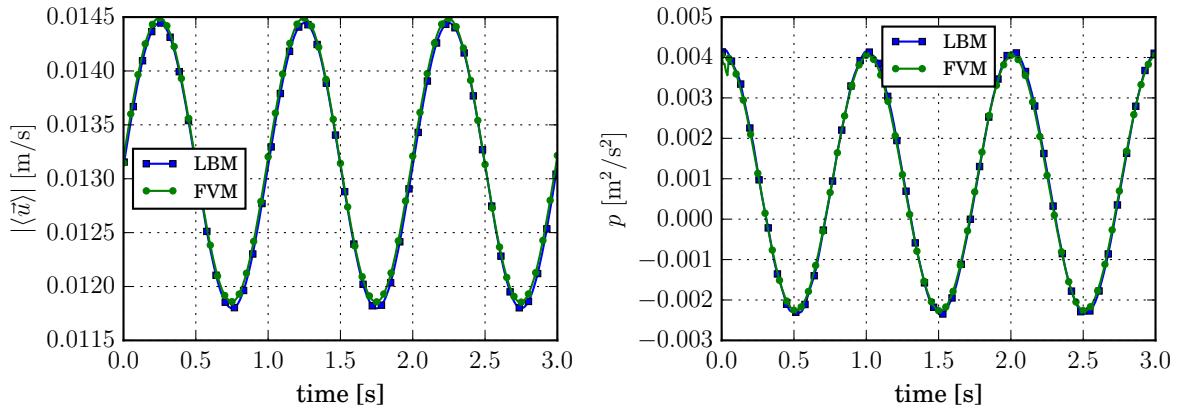


Figure 5.9: Profile of the average velocity magnitude $|\langle\vec{u}\rangle|$ (left panel) and pressure p (right panel) on the middle slice of the U-shape domain in an oscillatory flow simulation. Data from LBM and FVM is compared in physical units. Density-normalized pressure is shown.

Performance Table 5.1 presents a short summary of the simulation performance for flows within the artificial U-shaped geometry. The FVM simulation times using a Quadro K6000 GPU with the same termination criterion ($\epsilon_\rho^\infty < 10^{-4}$, $\epsilon_{\vec{u}}^\infty < 10^{-4}$) are 317 s (stationary Re = 100), 663 s (stationary Re = 1000), and 4900 s (oscillatory $\mathrm{Re}_{\mathrm{avg}} = 100$, 3 cycles with 1 discarded to avoid transients).

We note that the stationary Re = 100 case is 11x faster in LBM, providing results in less than half a minute from the simulation start. The stationary Re = 1000 case is only 1.2x faster, while the oscillatory $\mathrm{Re}_{\mathrm{avg}} = 100$ case is 3.7x faster.

In general, LBM, being a fully dynamic method, is inefficient for stationary simulations. Modifications to the core LB algorithm are possible to accelerate convergence though [145] — something we have not explored here since dynamic simulations were of primary interest to use in the context of hemodynamics. In the case of the oscillatory flow, the combination of Re, α, and a domain with a relatively long flow path distance between the inlet and the outlet, forced us into a regime of low fluid speeds, which in turn necessitated the use of round-off minimization model for the simulations, as well as wasting 75% of the total simulation time on waiting for initial transients to disappear.

5.3.4 Internal Carotid Artery

The internal carotid artery (ICA) located in the human neck and head supplies the brain with oxygenated blood. For simulations, we used a fragment of the C0006 geometry from the Aneurisk database containing an aneurysm (see Figure 5.10). The inlet radius for this geometry is $r = 1.99$ mm. We simulated a velocity-driven flow with $\mathrm{Re} \in (343, 810)$

Table 5.1: LB simulation performance for the U-shape case on a Quadro K6000 GPU. All simulations used single precision. The execution time does not include the setup phase (20 - 45s). For stationary simulations time until $\epsilon_\rho^\infty < 10^{-4}$, $\epsilon_{\vec{u}}^\infty < 10^{-4}$ is shown.

Case	Domain size (active nodes)	MLUPS	Iterations	Wall time [s]
stationary $\mathrm{Re} = 100$	$1.2 \cdot 10^6$	778	$1.6 \cdot 10^4$	28
stationary $\mathrm{Re} = 1000$	$4.98 \cdot 10^6$	765	$8.8 \cdot 10^4$	556
oscillatory $\mathrm{Re}_{\mathrm{avg}} = 100$	$1.2 \cdot 10^6$	807	$7.86 \cdot 10^5$	1181

and $\alpha = 5.84$, with a time-dependent inlet velocity following a physiological profile [36] (see Figure 5.11).

For the FVM simulation, the input STL file was used as a base for a volumetric mesh consisting of $3.21 \cdot 10^5$ tetrahedral elements, and generated using the NetGen software. The LBM simulation used the D3Q19 lattice with $1.36 \cdot 10^6$ active fluid nodes located in a bounding box of $217 \times 301 \times 171$. On this lattice, the inlet diameter was resolved with 52 nodes. The incompressible LBGK model was used for relaxation with $u_0 = 0.025$. No special initialization procedure was used before the main simulation, but the first cycle of the inflow waveform was discarded to allow for initial transients to disappear.

Results Figure 5.12 presents a comparison of average velocity and pressure profiles measured in the middle of the domain in the y direction. A good overall agreement is seen. While the FVM solution was completely laminar, resulting in a smooth profile of the average velocity and pressure, the LBM simulations showed unsteady flow, with vortices appearing in the area of the first bend of the artery.

The unsteadiness of the solution makes the average velocity curve appear jagged. This is to be expected, as vortices are being shed with a frequency much higher than that of the driving waveform, and as they move in the 3-dimensional space through the 2D slice on which measurements were taken, they lead to a seemingly random noise being imposed on an overall smooth profile. This can be also seen by taking a moving window temporal average of the velocity time series. We did that by averaging the nearest 10 data points — the result is presented as the „LBM smoothed" curve on Figure 5.12.

The smoothed average velocity profile can be seen to be in very good agreement with FVM, indicating that all large-scale flow field structures are well resolved. Pressure on the other hand seems to be overpredicted by up to 25% in the LBM simulations, in comparison to the FVM results.

Unsteady flow analysis In order to understand and validate this result, we performed a large number of additional simulations. First, to simplify the problem, we switched the inflow boundary condition to use a constant paraboloid profile corresponding to maximum Re attained in the original simulation (i.e. velocity from the systole peak). We then verified that the flow unsteadiness is also visible with different relaxation models (MRT, ELBM), different maximum inflow fluid velocity ($u_0 = 0.01, 0.005$), double precision, different inflow/outflow boundary conditions (regularized, equilibrium) as well as with a lattice 2x larger in every direction (8x more fluid nodes). We performed the same simulation with Palabos [112], another open source package implementing the LBM, obtaining matching results. We also experimented with artificially extending the inlet branch and removing the aneurysm sac and the outlet branch in order to see if the unsteadiness is coupled with flow conditions near the inlet/outlet. These experiments yielded a negative result, i.e. the flow character remained unchanged.

After elimination of these possible causes of the instability, we looked at the Fourier spectra of velocity and density time series extracted from various points in the domain, both on the middle y slice and from down- and upstream locations. Around the slice, the spectra were sharply peaked (sometimes including harmonics), suggesting that the instability is caused by a regular vortex shedding process (which was also confirmed with visual observation of the flow field during the simulation). We noted that close to the inlet (after the aneurysm) the flow field appeared highly disturbed visually, but the spectra were flat with no peaks. This suggests that the fluid is well mixed in the aneurysm sac and it might be getting close to the weakly turbulent regime near the outlet.

We then tried to reproduce these results with a constant inflow profile with FVM. We used a time-dependent solver (PISO) with a time step of $5 \cdot 10^{-6}$ s. The obtained solution was also periodic, but vortex shedding occurred only close to the outflow part of the previously analyzed y-slice and was relatively weak (which explains why no „noise" is visible in FVM results in Figure 5.12), whereas in LBM simulations vortex shedding was present directly after the first bend of the ICA so that vortices moved through the majority of the slice area.

Finally, we performed a series of simulations aimed at establishing when the instability starts to occur. To do this, we gradually increased the inflow velocity over $5 \cdot 10^5$ iterations from 0 to the target value corresponding to $\mathrm{Re} = 800$. This analysis revealed that instability in the part of the vessel leading into the aneurysm (i.e. after the second bend) starts to occur around $\mathrm{Re} \approx 280$, while vortex shedding after the first bend appears at $\mathrm{Re} \approx 580$.

While blood flow in most human arteries is generally considered to be non-turbulent, literature analysis shows that the existence of disturbed flow in the ICA is not entirely unexpected. Fischer et al. [35] performed simulations showing that weakly turbulent flow can develop close to the ICA bifurcation, at Reynolds numbers similar to the one

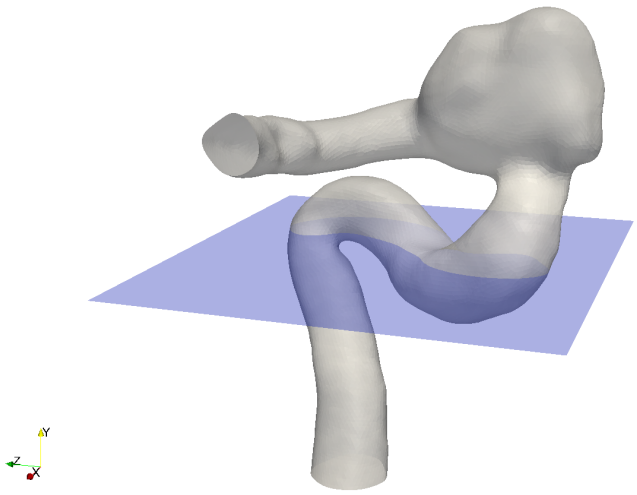


Figure 5.10: ICA geometry, extracted from the C0006 case from the Aneurisk database. The inflow plane where velocity boundary conditions are applied is located at the bottom of the image. The blue plane shows where measurements for comparison with the FVM simulation were taken.

used in our case. Závodszky and Paál [160] in their comparison of FVM and LBM in ICA flow noted that LBM better reproduced experimental measurements and FVM significantly underpredicted velocity at the systolic peak. They speculated that in a complex geometry like that of the ICA, the flow might not stay in the laminar regime throughout the full cardiac cycle. Kerber and Heilman [71] discussed experimental data obtained from flows in a transparent model of real human artery. They noted that in the petrous ICA laminar flow patterns change into helical ones, which continues into the cavernous ICA. Localized flow reversal during systole was also seen, with the overall flow being called „highly disturbed" but not turbulent [71]. Another experimental result was presented by Schubert et al. [124], who observed helical flow structure through magnetic resonance imaging of ICA blood flow in healthy humans.

Flow field structure In order to better understand the structure of the flow field over the course of one cardiac cycle, we plotted streamlines and Q-criterion [65] isosurfaces at times of maximum and minimum inflow speed (see Figure 5.13). Streamlines were calculated starting from a sample of 100 points located close to the inlet.

The flow close to the inlet can be seen to be completely laminar, as could be expected based on the boundary conditions that we set. We note that overall the geometry has three zones where the flow is accelerated: the 180 degree bend, the „neck" of the aneurysm, and the part of the artery leading outside of the aneurysmal sac. This local

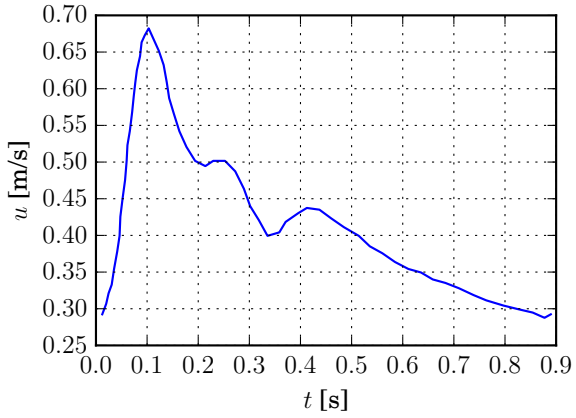


Figure 5.11: Inlet velocity waveform for the internal carotid artery simulation [36].

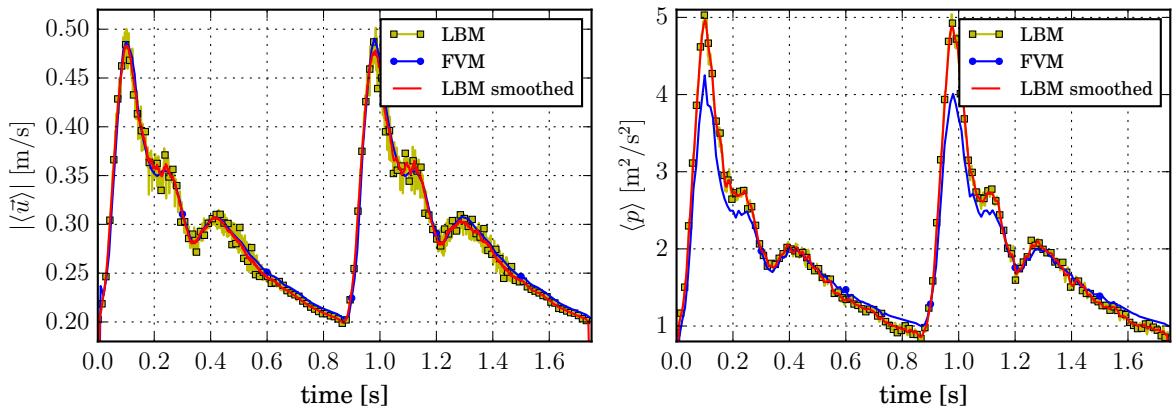


Figure 5.12: Profile of the average velocity magnitude $|\langle\vec{u}\rangle|$ (left panel) and pressure p (right panel) on the middle y-slice of the internal carotid artery simulation. Data from LBM and FVM is presented scaled to physical units, and density-normalized pressure is shown.

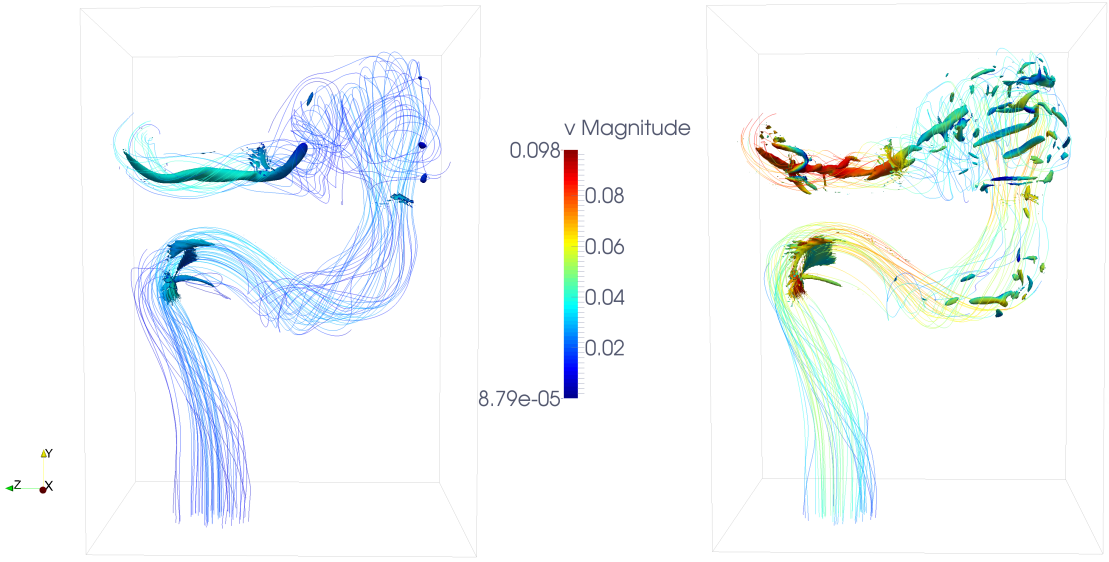


Figure 5.13: Streamlines at moments of lowest inflow velocity $t = 0\,\mathrm{s}$ (left panel) and highest inflow velocity $t = 0.1\,\mathrm{s}$ (right panel). Isosurfaces of the Q value are shown to highlight areas where vortical structures develop. Both streamlines and isosurfaces are colored according to the local velocity, which is shown in lattice units (mean inflow speed was $u = 0.025$).

acceleration is present regardless of the phase of the cardiac cycle and is driven by physiological narrowing of the vessel lumen.

In the first bend, the fluid is driven towards the outward wall by inertia, which has a flattening effect — if we consider a bunch of streamlines uniformly distributed within a circle close to the outlet, they will form a sheet structure close to the bend. This also leads to the formation of a stagnation zone at the opposite wall. Slow helical flow can be seen to form in this zone (this is not visible in Figure 5.13 since the location of the sampling points for the streamlines close to the inlet means that the vast majority of them pass through the sheet structure and do not cross the stagnation zone). The presence of two layers of fluid moving at significantly different velocities leads to shearing forces in this region, which could be the source of the previously discussed oscillations through a Kelvin-Helmholtz instability.

As the fluid moves towards the next bend, it starts to acquire a helical structure as the sheet hits the bottom part of this section of the artery and is accelerated upwards in the direction of the aneurysmal sac. In the sac, blood impinges on the rear wall and starts to swirl so that upon leaving the sac a fully developed helical flow can be seen in the branch of the artery close to the outlet.

To better visualize the helical parts of the flow, we computed isosurfaces of Q, defined

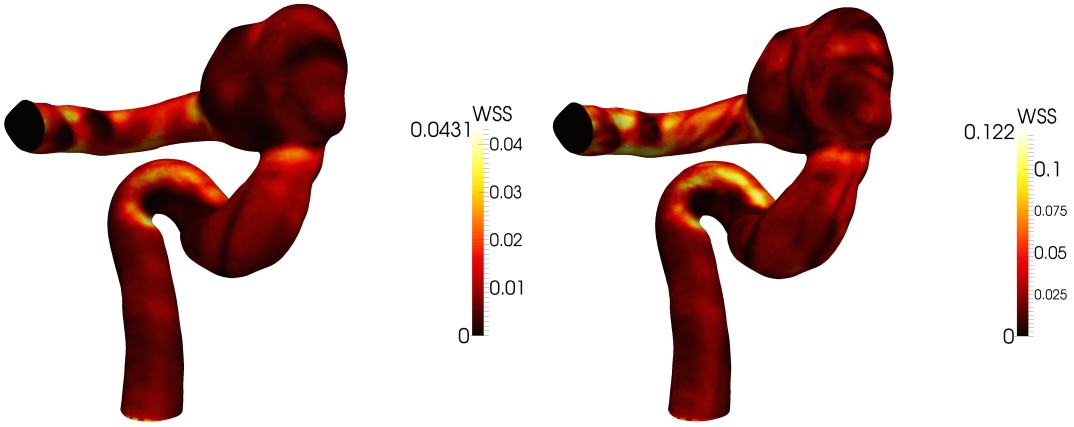


Figure 5.14: WSS in the ICA at diastole (left panel) and systole (right panel). WSS value is specified in Pascals.

as:

$$Q = \frac{1}{2}\left(\partial_\alpha u_\beta \partial_\beta u_\alpha\right) = \frac{1}{2}\left(|\Omega|^2 - |S|^2\right)$$
$$S_{\alpha\beta} = \frac{1}{2}\left(\partial_\alpha u_\beta + \partial_\beta u_\alpha\right) \qquad \Omega_{\alpha\beta} = \frac{1}{2}\left(\partial_\alpha u_\beta - \partial_\beta u_\alpha\right).$$

Q can be seen to measure the balance between shear strain rate and vorticity magnitude.

The Q isosurfaces at two different representative levels are shown in Figure 5.13, with the level for the right panel being 6.25 times higher than that for the left one. By far the most dominant helical structure is the one formed after the aneurysm, which is clearly visible at both diastole and systole. We also note the presence of a small isosurface tube in the previously mentioned stagnation zone. At systole, a lot of new structures appear in the region of the second bend and primarily within the aneurysmal sac, indicating the formation of vortices in these areas.

WSS analysis In order to compute the WSS for the simulated flows we applied the geometric method [99] to establish normal vectors for the used geometry. We experimented with different sizes of the averaging neighborhood (5^3, 11^3 and 21^3), as well as exponents in the weighting function $(1+r)^{-\gamma}$, where r is the distance between the weighted voxel and a center voxel. We found that the different options had minimal impact on the results, so we proceeded with the rest of the analysis using a kernel size of 5^3 and $\gamma = 1$. The normal vectors established with this procedure were then used to compute the WSS vector according to Eq. (5.3), with the stress tensor computed in the simulation using Eq. (5.1). The cardiac cycle was sampled with 100 regularly spaced data points, which were then used to compute the OSI using Eq. (5.4).

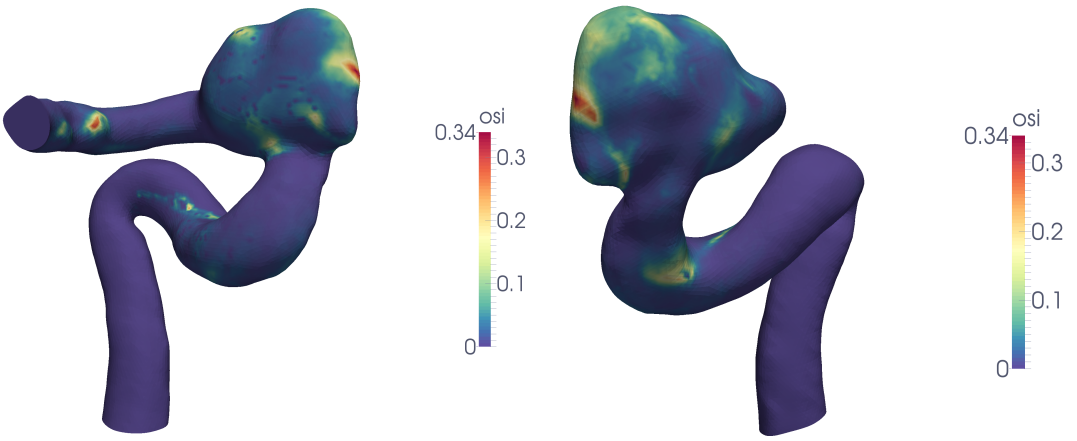


Figure 5.15: OSI in the ICA. Left panel: front view, right panel: rear view.

For purposes of visualization, we applied the grey dilation morphology operation with size 3^3 on the OSI and WSS fields. As the dilation operation effectively performs max-sampling within a neighborhood of the central voxel, its application reduces the spatial resolution by up to two lattice units. Since we were interested in visualization only, this was deemed acceptable. The processed data was then visualized after the application of the `ResampleWithDataset` filter in the Paraview software. The filter was used to map the field values from the LB lattice back onto the surface geometry which was used to set the boundary conditions of the simulation. The results are presented in Figure 5.14 and Figure 5.15.

Figure 5.14 shows that the basic distribution of WSS stays the same throughout the cardiac cycle, with the magnitude changing by a factor of ~ 3 between diastole and systole. Areas of elevated WSS include the first bend of the vessel, the neck of the aneurysm, and the outlet branch. All these areas are associated with faster flow (cf. Figure 5.13), and locations where a fluid jet impinges upon the vessel wall. Notable areas of low WSS, which are more important from a clinical standpoint, are the inner side of the first bend, and a bulge on the outlet branch. In both of these locations previous analysis showed regions of slow-moving fluid.

The OSI (see Figure 5.15) provides a complementary picture. High OSI indicates areas where the direction of the $\vec{\tau}_w$ varies over the cardiac cycle. The bulge on the outflow branch shows high OSI, providing additional evidence that this location is at risk of wall weakening. Other areas of high OSI include the aneurysm sac, where the flow becomes highly disturbed during systole and parts in the middle of the geometry associated with formation of vortices.

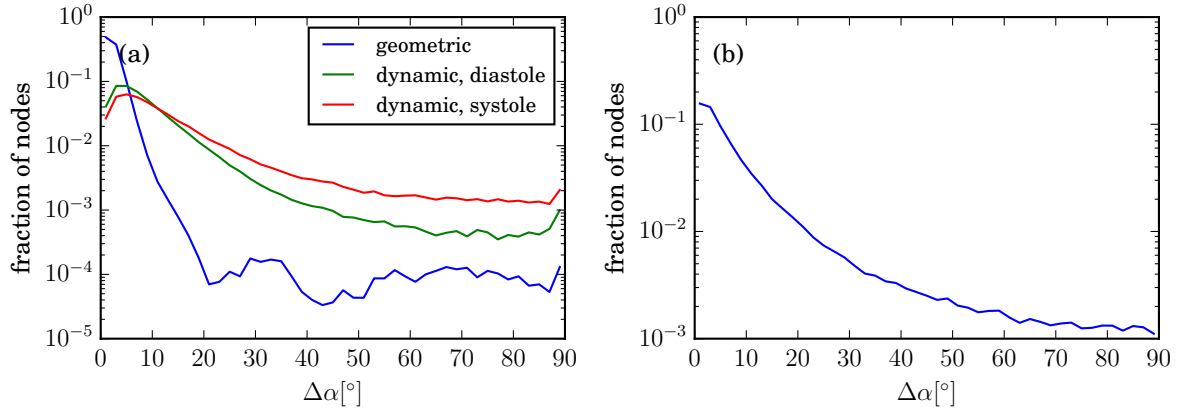


Figure 5.16: Distributions of angles between normal vectors computed for a layer of 3 nodes next to the wall of the ICA geometry with various methods. Panel (a) shows differences between nearest-neighbor normals from the underlying surface geometry (see text) and results from the geometric and dynamic method. Panel (b) shows differences between the results of the dynamic method applied at systole and at diastole.

Normal vectors for WSS To additionally validate our WSS results, we also applied the two other methods of computing the normal vector — the dynamic method [133], as well as a method based on using the normal vector of the underlying surface geometry directly. In the latter method, we associated the normal vector of every facet from the STL file with the centroid of the facet. Then, for every node of the LB lattice laying no farther than 3 lattice units from a wall, we identified the 5 nearest facet centroids, and averaged their normal vectors with a weight proportional to $1/d$, where d is the Euclidean distance between the node location and the facet centroid.

The results of all three methods are compared in Figure 5.16. In the left panel, we took the externally provided normal vectors as „ground truth" and for every near-wall node computed the angle between the ground truth vector and a normal vector obtained using either the geometric or the dynamic method.

In this test, the geometric method seems to work significantly better — 99% of the analyzed nodes show a normal vector deviation from ground truth no larger than 10° compared to only 33% with the dynamic method at diastole and 25% at systole. The dynamic method also results in different normals depending on the phase of the cardiac cycle at which the evaluation happens. The right panel in Figure 5.16 shows discrepancies between the normal vector direction at systole and diastole. Only 51% of the nodes show a normal vector change of less than 10°.

To further evaluate the practical impact of the differences seen in Figure 5.16, we also computed WSS using each set of normal vectors and compared them all against WSS

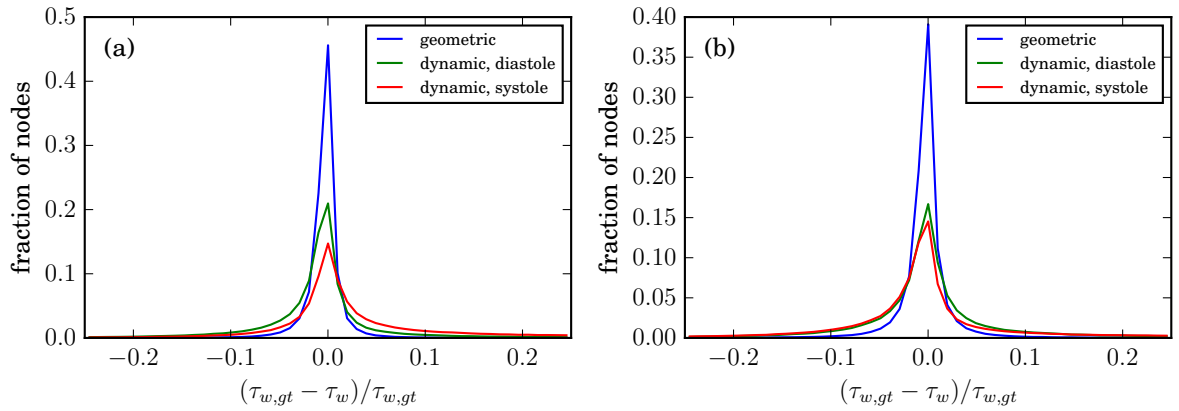


Figure 5.17: Relative changes in the length of $\vec{\tau}_w$ between results using normal vectors computed with the geometric and dynamic methods, and ground truth normals from the STL file of the ICA geometry ($\vec{\tau}_{w,gt}$). Left panel: diastole, right panel: systole.

computed using the ground truth normal vectors from the STL file defining the geometry. The results of this comparison at both diastole and systole are shown in Figure 5.17 in panels (a) and (b), respectively. The geometric method again performs best here, having the most sharply-peaked distribution of relative errors. The error distribution of the dynamic method is more heavy-tailed, with some errors larger than 10%, and changes visible between the normals computed at diastole and systole.

A visual comparison of the 3D data in ParaView (not shown here) shows that the spatial distribution of WSS magnitude over the walls of the vessel is very similar across all four sets of normal vectors. Only minor local changes visible, which are not expected to have a significant impact on the usefulness of the results in a clinical setting. All methods of normal vector estimation could therefore be considered to provide data of acceptable quality. Given the relative complexity of the dynamic method (necessity of solving an eigenvalue problem for every node where the normal vector is to be computed; unclear at which point in a transient flow to apply the computation), we conclude that the geometric method or, where available, externally-provided normal vectors, should be preferred.

Performance The LBM simulation ran until a full 3 cycles of the inflow waveform were complete, which corresponded to $9.38 \cdot 10^5$ iterations. The average solver speed was 490 MLUPS and the simulation took 2605 s in single precision on a Quadro K6000 GPU. The FVM simulation of the same case took 76278 s on a Tesla M2075 GPU. LBM thus provided a 22x acceleration (the actual speedup factor might be slightly lower due to speed differences between the two GPU used for the comparison).

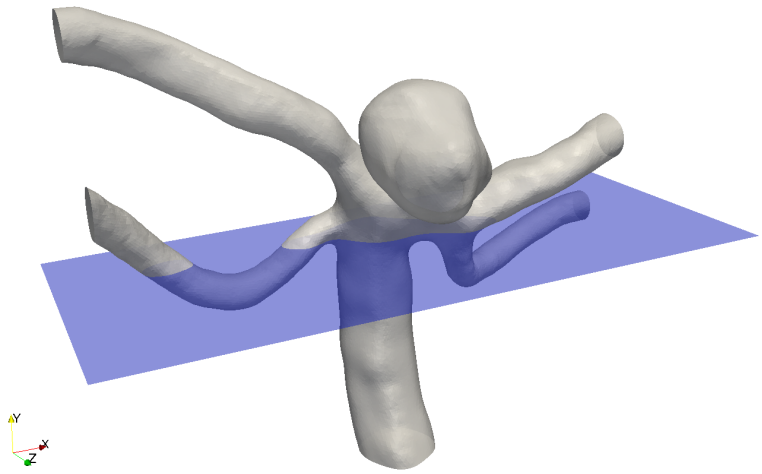


Figure 5.18: BA geometry extracted from the C0032 case from the Aneurisk database. The inflow plane is located at the bottom of the image. Other branches are used as outflows. The blue plane shows where measurements for comparison with the FVM simulation were taken.

5.3.5 Basilar Artery

The basilar artery (BA) is one of the blood vessels supplying the human brain with oxygen-rich blood. Similarly to the previously discussed ICA, the BA is part of the circle of Willis — a cerebrovascular structure providing blood to the brain. For simulations, we used a fragment of the C0032 geometry containing an aneurysm (see Figure 5.18). The inlet radius for this geometry is $r = 1.68$ mm. Similarly to the ICA case, we simulated a velocity-driven flow with $\mathrm{Re} \in (348, 616)$ and $\alpha = 4.63$. The waveform of the velocity followed a physiological profile [143] (see Figure 5.19).

The FVM simulation used a volumetric mesh consisting of $8.29 \cdot 10^5$ tetrahedral elements, generated using the NetGen software. The LBM simulation used the D3Q19 lattice with $1.69 \cdot 10^6$ active fluid nodes located in a bounding box of $179 \times 354 \times 397$. On this lattice, the inlet diameter was resolved with 66 nodes. The incompressible LBGK model was used for relaxation, with $u_0 = 0.05$. Further increases of the lattice size did not improve the solution. No special initialization procedure was used before the main simulation, but the first cycle of the inflow waveform was discarded to allow for initial transients to disappear.

Figure 5.20 presents a comparison of average velocity and pressure profiles, measured in the middle of the domain in the y direction. A very good agreement can be seen with FVM results, with only a slight difference in the peak pressure.

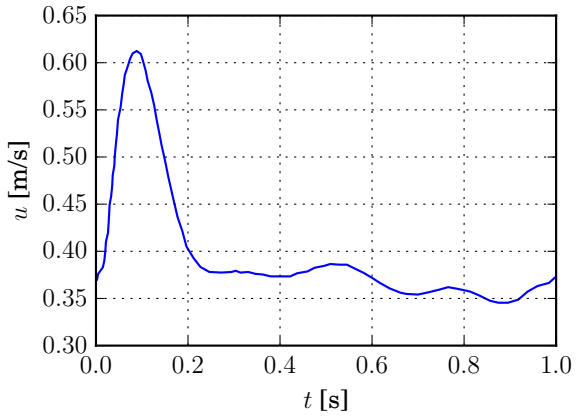


Figure 5.19: Inlet velocity waveform for the basilar artery simulation [143].

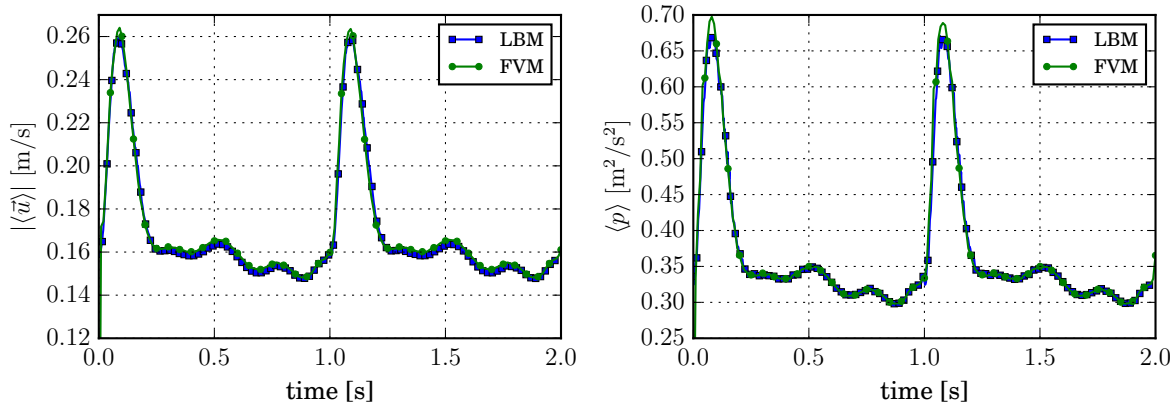


Figure 5.20: Profile of the average velocity magnitude $|\langle\vec{u}\rangle|$ (left panel) and pressure p (right panel) on the middle y-slice of the basilar artery simulation. Data from LBM and FVM is presented scaled to physical units, and density-normalized pressure is shown.

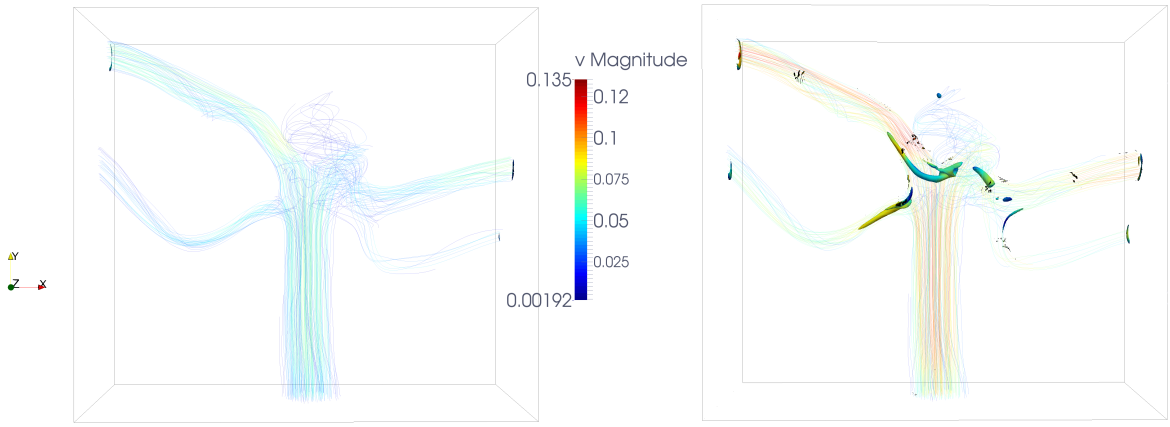


Figure 5.21: Streamlines at moments of lowest inflow velocity $t = 0\,\mathrm{s}$ (left panel) and highest inflow velocity $t = 0.1\,\mathrm{s}$ (right panel). Isosurfaces of the Q value are shown to highlight areas where vortical structures develop. Both streamlines and isosurfaces are colored according to the local velocity, which is shown in lattice units (mean inflow speed was $u_0 = 0.5$).

Flow field structure The structure of the velocity field in the case of the basilar artery is simpler than in the case of the previously discussed ICA, mainly due to lack of sharp bends in the geometry. We took the same approach as in the case of ICA, and plotted streamlines and isosurfaces of Q value at systole and diastole (see Figure 5.21).

The flow can be seen to remain laminar throughout the cardiac cycle, with some weakly helical flow present around the aneurysmal sac at systole. The majority of the inflow momentum is carried through the two upper outlet branches, owing to their larger diameters. Small vortices develop close to the beginning of the upper branches, as well as in the neck of the aneurysmal sac. The velocities in the sac remain significantly lower than those in the branches throughout both systole and diastole.

WSS analysis To analyze the WSS distribution in the basilar artery case we applied the methodology discussed in Section 5.3.4. Similarly to the ICA, the overall distribution of relative WSS magnitude does not change much between systole and diastole, but the absolute shear stress values are higher at systole (see Figure 5.22). High WSS values can be observed in the area where the main inlet branch splits into four outlet branches and the aneurysm sac, as well as along the two upper outlet branches. Areas of low WSS include most of the aneurysm sac itself, and some sections of the outlet branches.

The OSI distribution (see Figure 5.23) again shows a good correlation between regions of high OSI and low WSS. Areas where wall weakening could be expected include part

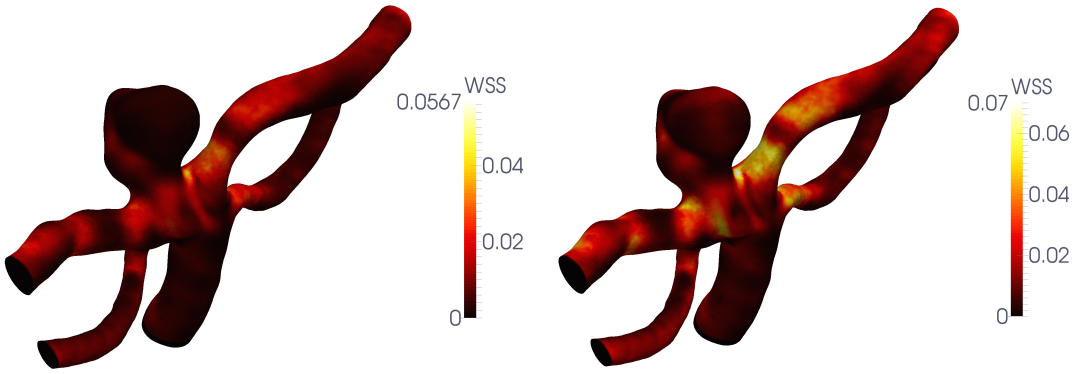


Figure 5.22: WSS in the BA at diastole (left panel) and systole (right panel). WSS value is specified in Pascals.

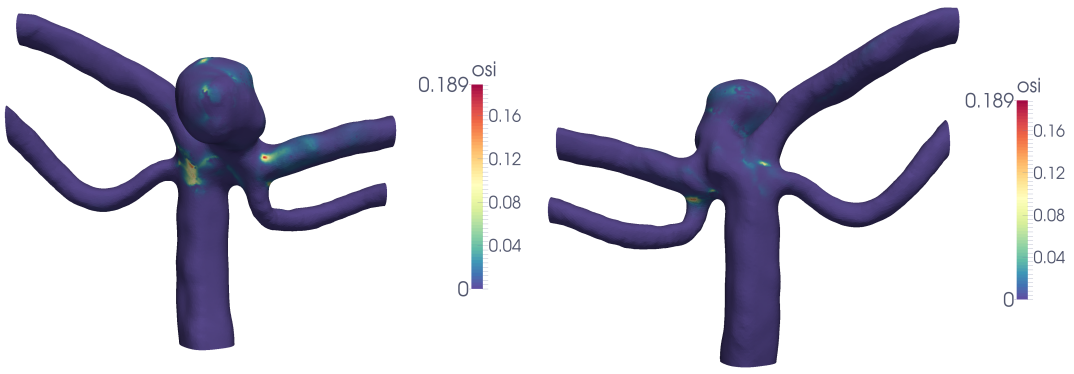


Figure 5.23: OSI in the BA. Left panel: front view, right panel: rear view.

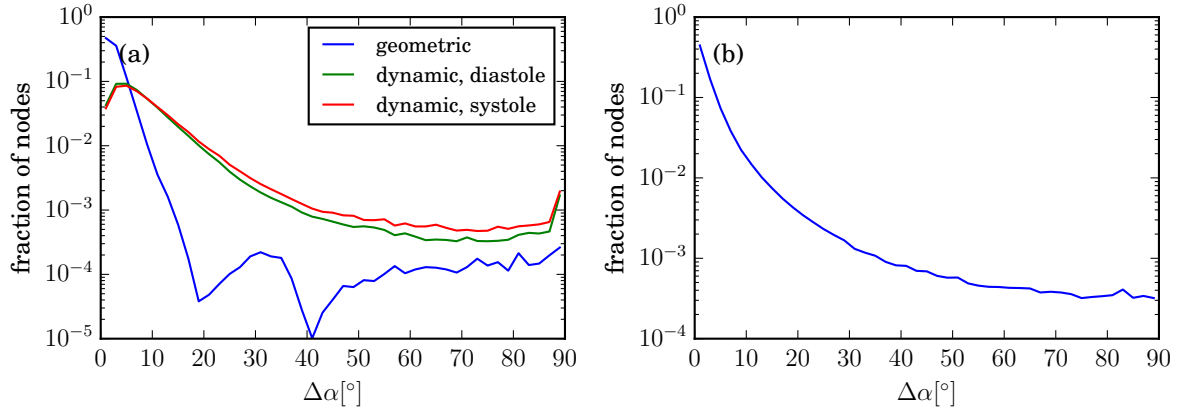


Figure 5.24: Distributions of angles between normal vectors computed for a layer of 3 nodes next to the wall of the BA geometry with various methods. Panel (a) shows differences between nearest-neighbor normals from the underlying surface geometry (see text) and results from the geometric and dynamic method. Panel (b) shows differences between the results of the dynamic method applied at systole and at diastole. Compare with Figure 5.16.

of the main inlet artery close to the beginning of the lower left outlet branch, localized spots near the beginning of the right outlet branches (associated with the development of vortices at systole — cf. Figure 5.21), as well as some areas of the aneurysm sac.

Impact of normal vectors We repeated the analysis of the impact of normal vector estimation in the case of the basilar artery, comparing both the normal vectors themselves (see Figure 5.24) and the WSS magnitude (see Figure 5.25). The conclusions we reached based on the ICA analysis also hold for the BA geometry — the geometric method consistently provides better results. In the BA case the flow is less disturbed that in the ICA and flow speed differences between diastole and systole are smaller, which results in better self-consistency between the normal vectors obtained with the dynamic method at diastole and systole (cf. Figure 5.24(b), Figure 5.16(b), and Figure 5.25).

Performance The LBM simulation ran until a full 3 cycles of the inflow waveform were complete, which corresponded to $7.14 \cdot 10^5$ iterations. The average solver speed was 319 MLUPS and the simulation took 3780 s in single precision on a Quadro K6000 GPU. The FVM simulation of the same case took 76278 s on a Tesla M2075 GPU. LBM thus provided a 20x acceleration, with no degradation in the solution quality (as in the case of the ICA simulation, the actual speedup might be slightly lower due to the different GPUs used in the simulations).

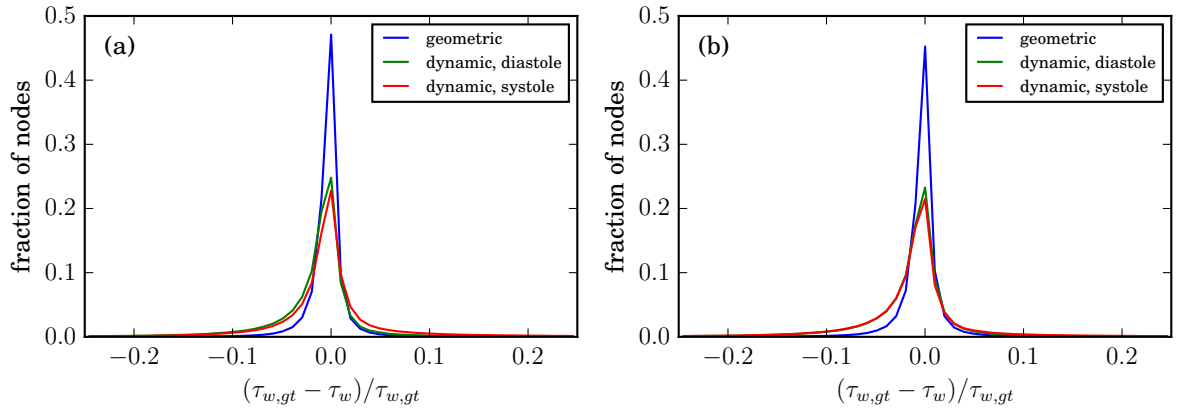


Figure 5.25: Relative changes in the length of $\vec{\tau}_w$ between results using normal vectors computed with the geometric and dynamic methods, and ground truth normals from the STL file for the BA geometry ($\vec{\tau}_{w,gt}$). Left panel: diastole, right panel: systole. Compare with Figure 5.17.

6 Turbulence

6.1 Turbulent flows

The term *turbulence* refers to a fluid flow regime associated with high Re, where the fluid motion is highly complex and chaotic. The unpredictable nature of the flow motivates a statistical treatment of its properties for practical engineering applications. The flow can in principle still be described by the NSEs, but the range of scales that have to be resolved in a numerical simulation quickly increases to a level where attempting to directly solve the motion equations is impractical due to the required computational power and memory. A common approach known as Large Eddy Simulation (LES) is to run the simulation at a reduced resolution and model the unresolved scales though a modification of the NSEs, e.g. as in the Smagorinsky subgrid model where an eddy viscosity is computed from local velocity gradients. If a simulation resolves all relevant spatial and time scales and attempts no subgrid modeling, it is called Direct Numerical Simulation (DNS).

In addition to being highly irregular, turbulent flows are *rotational*, having generally non-zero vorticity, and *dissipative*, needing an external energy source to sustain the turbulence. Their flow structure can be described in terms of *eddies* — coherent patterns of velocity, vorticity and pressure, which are formed at many length scales. Most of the kinetic energy is stored in large-scale eddies, and as they are broken up into smaller ones, a cascade is formed transporting energy to smaller-scale structures. This transport can be approximated as inertial and inviscid, so energy is transported but there is little dissipation taking place until a small enough scale is reached. Then, molecular diffusion starts playing a role and the kinetic energy is lost as heat due to viscous dissipation. This smallest scale is referred to as the *Kolmogorov scale*.

Kolmogorov formulated what is now known as the K41 theory [12, 77, 138]. While large-scale eddies can easily be seen to be anisotropic, Kolmogorov proposed that at small enough scales eddies become homogeneous and isotropic. This was stated in the form of two hypotheses:

1. At the small isotropic scales, the statistics of motion are uniquely determined by two parameters: the average energy dissipation rate per unit mass ϵ, and the kinematic viscosity ν.
2. At inertial scales (scales between the small scale on which dissipation occurs and the large scale of eddy motion), viscosity becomes irrelevant and statistics of motion

are determined only by ϵ.

The K41 theory is currently understood to be only approximately correct [12], due to the small-scale activity in turbulent flows being „clumpy", which breaks the K41 assumption of self-similarity. This pheonomenon is called „intermittency", in reference to the irregularity of kinetic energy dissipation.

For the purposes of further discussion, we now introduce notation for the two main length scales relevant for turbulent flows:

- the *integral* length scale L, proportional to the size of the largest eddies, which is determined by the size of the physical system,
- the *Kolmogorov* length scale $\eta = \left(\nu^3/\epsilon\right)^{1/4}$, which is the smallest hydrodynamic length scale for turbulent flows.

The integral length scale can be used to define an associated time scale, called *large eddy turnover time* $t_{\text{eddy}} = L/U$, where U is the characteristic speed of the flow. The Kolmogorov length scale can be arrived at through dimensional analysis of the two relevant driving factors. The dissipation rate can be estimated as $\epsilon \propto U^2/t_{\text{eddy}}$ by taking into account that it has to be equal to the kinetic energy production rate in a statistically steady flow. Taken together, the ratio of the integral to Kolmogorov lengths can be shown to scale as $L/\eta = \text{Re}^{3/4}$. At this growth rate, DNS quickly becomes infeasible as Re increases, which demonstrates the need for turbulence modeling.

In discussion of turbulent flows, the *Reynolds decomposition* is often used, where a quantity such as a fluid velocity component $u_i(t)$ is treated as a random variable and split into a mean and fluctuating component: $u_i(t) = \overline{u}_i + u_i'(t)$, where the fluctuating component has an mean value of $\overline{u_i'} = 0$ and where $\overline{x}$ indicates a time average of the quantity x. When this decomposition is substituted into the NSE for an incompressible Newtonian fluid, the resulting equations are known as Reynolds-Averaged Navier-Stokes equations (RANS):

$$\rho\overline{u}_j\partial_j\overline{u}_i = \rho\overline{f}_i + \partial_j\left(-\overline{p}\delta_{ij} + \mu\left(\partial_j\overline{u}_i + \partial_i\overline{u}_j\right) - \rho\overline{u_i'u_j'}\right). \tag{6.1}$$

The last term is known as the *Reynolds stress tensor*:

$$\tau_{ij}' = \rho\overline{u_i'u_j'} \tag{6.2}$$

which has to be modeled if the RANS are to be used for simulations. The Reynolds decomposition can also be used to define a number of other useful quantities, which we will later use to analyze simulation results:

- the n-th central moment of u: $\overline{u'^n} = \overline{(u-\overline{u})^n}$,
- the turbulence intensity: $\sqrt{u'^2}$,
- skewness: $\overline{u'^3}/\overline{u'^2}^{3/2}$ (determines the symmetry of the distribution function),

- kurtosis/flatness: $\overline{u'^4}/\overline{u'^2}^2$.

From the point of view of applications, turbulence is important for two main reasons. First, it enhances the mixing and transport of energy and matter, reaching much higher diffusion rates than those of molecular diffusion. This can be exploited for beneficial purposes, e.g. for fuel and oxidizer mixing in the case of combustion. Second, turbulence can lead to significant energy loss in fluid flows, both external and internal. Reducing the size of turbulent regions, or delaying transition from laminar to turbulent flow, can therefore lead to significant energy savings.

In this chapter we will mainly be concerned with the weakly turbulent regime, where it is still practical to attempt DNS simulations. The results of these fully resolved simulations will serve as ground truth against which lower resolution simulations will be compared.

6.1.1 Boundary layer

While a fully free turbulent flow could be considered inviscid, the presence of a wall creates a boundary layer where viscous effects are always important regardless of Re. If we denote the thickness of this layer with δ and the wall-normal distance with y, we can further subdivide the boundary layer into three regions [132]:

- $y/\delta < 0.025$: the buffer layer, including a thin viscous sublayer,
- $0.025 < y/\delta < 0.2$: the log region,
- $y/\delta > 0.2$: the wake region.

We will now discuss the idealized case of flow in the presence of a smooth, semi-infinite plate, which is the simplest possible wall-bounded turbulent flow geometry. This setup, while simple, nevertheless has useful properties (some of which are considered to be universal) and as such is useful as a basis for description of more complex settings.

If we denote the kinematic wall shear stress by:

$$\tau_w = \nu \left.\frac{\partial u}{\partial y}\right|_{y=0}, \tag{6.3}$$

with u being the velocity component tangential to the wall, it will be convenient to define a reference *wall friction* velocity $u_\tau = \sqrt{\tau_w}$. A system of units normalized by this velocity will be called *wall units*, and denoted with the subscript $+$. In this system, the wall-normal distance is $y_+ = yu_\tau/\nu$ and the fluid streamwise velocity is $u_+ = u/u_\tau$. The wall friction velocity also makes it possible to define an associated Reynolds number $\mathrm{Re}_\tau = u_\tau H/\nu$.

In a fully developed turbulent flow, away from the wall the Reynolds stress $\overline{uv}$ is significantly larger than the viscous stress. Close to the wall the relation is reversed, and it can be taken as the defining feature of the viscous sublayer. If we completely neglect

the Reynolds stress contribution in this region, the velocity variation in the sublayer can be shown to be linear with the distance from the wall $u_+ = y_+$. Experimental data shows the span of the viscous sublayer extends to $y_+ \approx 5$. Beyond the viscous sublayer, viscous dissipation remains important until a distance of the order of 30 wall units defining the viscous layer region. The part of the viscous layer beyond the viscous sublayer is often called the buffer layer ($5 \leq y_+ \leq 30$). It is in the middle of this region that rates of turbulent energy production $\overline{uv}\partial_y\overline{u}$ and dissipation reach a peak, corresponding to a location where the viscous and turbulent stresses are of the same magnitude. The viscous sublayer has been observed to have little dynamic function and to not be capable of effectively transporting momentum. Experiments showed that if wall roughness is contained within the viscous sublayer, it has very little effect on the mean velocity distribution in the rest of the flow [132]. As such, the viscous sublayer can be considered as an effective modified boundary condition for the rest of the flow.

Outside of the viscous layer, momentum transport is accomplished primarily by turbulent eddies. The size of this region grows with increasing Re and its characteristic feature is its approximately constant Reynolds shear stress, which is why the region is sometimes called the *constant stress region*. Given the dominance of turbulent transport in this layer, we can assume that the mean velocity in this region depends only on the Reynolds stress. If we also assume that the distance from the wall y is the only spatial scale that matters in this region, dimensional analysis leads to the relation

$$\frac{\partial \overline{u}}{\partial y} = \frac{1}{\kappa}\sqrt{\tau}/y \tag{6.4}$$

where κ is called the von Karman constant which is presumed universal. Experimental data shows that $\tau \approx \tau_w$ in this region. Eq. (6.4) can thus be integrated and written as:

$$u_+ = \frac{1}{\kappa}\log y_+ + B \qquad y_+ \gg 1 \tag{6.5}$$

where B is another constant. B is known to depend on geometry and wall roughness [96]. Commonly used empirical values for both of these constants are $\kappa = 0.41$ and $B = 5.5$. Eq. (6.5) is called the „log-law of the wall".

As the dynamics of the fluid in the constant stress layer is controlled almost exclusively by the Reynolds stress, dimensional analysis similar to that used to derive the log-law can lead to the conclusion that in this layer:

$$u' = \text{const} \qquad v' = \text{const} \qquad w' = \text{const} \qquad \overline{\epsilon} = \overline{p} = \frac{u_\tau}{\kappa y} \tag{6.6}$$

where p is the energy production rate.

The simple picture painted so far rests on the assumption that the wall acts a sink of momentum and that interactions are limited to neighboring fluid layers. One class of

phenomena known to occur in turbulent flows and clearly not captured by this theory are *intermittency* effects, where flow quantities have higher variability in space and time as would otherwise be expected.

Wall-bounded turbulent flows are also known to exhibit certain structural elements. For instance, long streaks form close to the wall in the streamwise direction [75]. These streaks appear most pronounced around $y_+ = 9$, seem to occur randomly in both space and time, but do have clear spatial coherence. Their mean spanwise spacing was measured to be about 100 wall units, which experimental data indicates to be invariant to the Reynolds number [74, 75, 130]. Dynamically, these streaks lead to so-called *bursting events*, which start with the formation of a streak, which is then lifted up from the wall region, starts to oscillate when it reaches $y_+ \approx 10$, and eventually breaks up in the buffer layer. Bursting events represent a mode of interaction between the inner and outer regions of the flow different than simple momentum flux, and significantly contribute to turbulent energy production [149].

6.2 Kida vortex

We begin by validating our code on the Kida vortex test case, which is a free decay from the initial state [72]:

$$\begin{aligned} u(\vec{x}, 0) &= u_0 \sin x \left(\cos 3y \cos z - \cos y \cos 3z\right) \\ v(\vec{x}, 0) &= u_0 \sin y \left(\cos 3z \cos x - \cos z \cos 3x\right) \\ w(\vec{x}, 0) &= u_0 \sin z \left(\cos 3x \cos y - \cos x \cos 3y\right) \end{aligned}$$

defined on a cubic domain with face length 2π, and periodic boundary conditions in all directions. The Kida vortex case is often used to validate numerical codes at high Reynolds numbers — a prerequisite for actually using them for turbulence simulations. The structure of the velocity field shows a high degree of symmetry — a feature that is sometimes exploited to significantly lower the memory and compute power requirements necessary to solve the problem at a given resolution. In our LB simulations we did not take advantage of this symmetry in any way, and instead simulated the whole system directly.

We performed simulations for $\mathrm{Re} = Nu_0/\nu = 4000$, $1.28{\cdot}10^4$, and $1.28{\cdot}10^5$ and compared them with results of Chikatamarla et al. [27] and Keating et al. [70]. The simulations were run using $u_0 = 0.05$ on a 700^3 grid (350^3 for $\mathrm{Re} = 4000$), using both single and double precision (with no noticeable difference between them). The $\mathrm{Re} = 4000$ and $\mathrm{Re} = 1.28 \cdot 10^4$ cases were investigated using the LBGK, MRT, regularized LBGK, Smagorinsky-LES and entropic models. At $\mathrm{Re} = 1.28 \cdot 10^5$, only simulations using the entropic model and the Smagorinsky subgrid model (with $C_S = 0.1$) remained stable.

During the simulation, kinetic energy

$$E = \frac{1}{2V} \int d^3x \vec{u}^2$$

and enstrophy

$$\Omega = \frac{1}{2V} \int d^3x (\vec{\nabla} \times \vec{u})^2$$

where V is the volume of the simulation domain, were tracked directly on the GPU. Vorticity was computed using the two-point second order central difference scheme in the bulk of the fluid and first order forward/backward differences at subdomain boundaries.

For Re = 4000 all four models gave the same results (Figure 6.1). At Re $= 1.28 \cdot 10^4$ some minor differences are visible, particularly in the evolution of enstrophy. Its peak value is slightly underpredicted by both models that locally modify effective viscosity (Smagorinsky, ELBM) (see Figure 6.1(c)). At Re $= 1.28 \cdot 10^5$, the differences are more pronounced and we observe that the Smagorinsky model underpredicts the absolute value of peak enstrophy. The kinetic energy spectrum shown on Figure 6.1(d) was computed as

$$E(k) = \sum_{k \leq k' < k+1} \hat{u}(k)^2 \qquad k = 0, 1, 2, \ldots .$$

where $\hat{u}(k)$ denotes the Fourier transform of the velocity field. A good agreement is visible in comparison to the Kolmogorov scaling $k^{-5/3}$, especially for the high Reynolds number cases. All collision models lead to similar spectra, with ELBM at Re = 4000 predicting a slightly higher value around $k = 10$ than LBGK or other models, and with ELBM keeping a slightly flatter spectrum for high k values at Re $= 1.28 \cdot 10^5$. In all cases the simulation results show the same features as those discussed in previous papers on this topic [27, 70, 72].

6.3 Turbulent channel

Channel flow is a benchmark problem which has been extensively studied for many years using both numerical and experimental methods. The geometry of the channel problem is that of two plates infinite in the streamwise (x) and spanwise (z) direction, placed at $y = 0$ and $y = 2H$, with H being the channel half-height (see Figure 6.2). Since a numerical simulation is always using a finite domain, the infinite span of the channel is simulated through periodic boundary conditions. This makes it necessary to introduce two additional parameters specifying the aspect ratios of the domain: $\alpha_x = L_x/H$ and $\alpha_z = L_z/H$, where L_x and L_z are the lattice sizes in the streamwise and spanwise directions, respectively. Jimenez and Moin [66] studied the impact of these aspect ratios and introduced the concept of a Minimal Flow Unit (MFU) corresponding to $\alpha_x = \pi$,

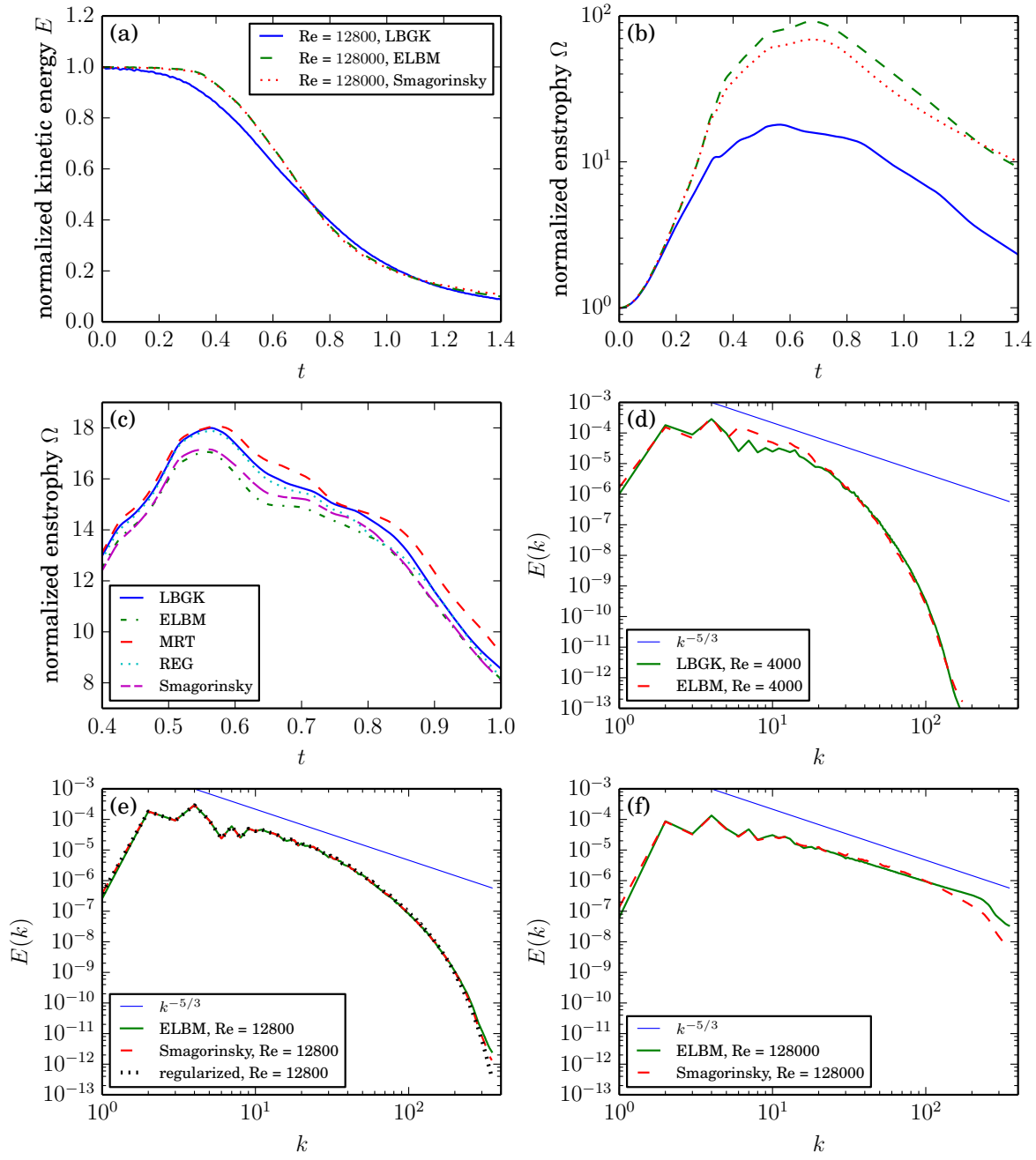


Figure 6.1: Kida vortex simulation results: (a) evolution of normalized kinetic energy, (b) evolution of normalized enstrophy, (c) evolution of normalized enstrophy at Re = 12800 for various collision models, (d-f) kinetic energy spectrum for selected collision models and Re = 4000, 12800, 128000, respectively. For panels (a-c) time is rescaled assuming a domain size of $(2\pi)^3$ and $u_0 = 1$. Panels (a) and (b) use the same line color coding.

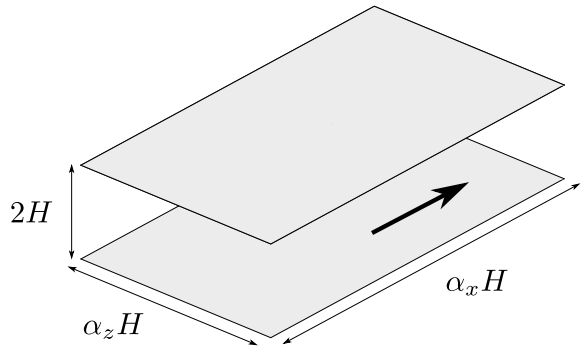


Figure 6.2: Geometry of the channel case. The big arrow indicates bulk flow direction.

$\alpha_z = 0.286\pi$. The MFU is large enough to sustain turbulence, but in practice larger domains are sometimes needed to get good profiles of mean velocities and Reynolds stresses, and clear minimum values are unknown for different numerical methods [148].

For statistical analysis of the simulation results it is useful to introduce the following two time scales. The *characteristic time* $t_c = H/u_\tau$ is the time taken by the slow flow in the viscous sublayer to traverse a distance of one channel half-height. The *flow-through time* $t_f = \alpha_x t_c$ corresponds to the time taken by the flow structures in the wall layer to traverse the whole simulation domain in the streamwise direction.

In order for a simulation to be considered a DNS, all relevant spatial scales need to be resolved. In practice, this means that the grid resolution Δ has to be comparable with the Kolmogorov scale η. For channel flows, η is expected to be smallest close to the wall and it has been estimated at $\eta_+ \approx 1.5$ [89].

6.3.1 Prior DNS results

Non-lattice Boltzmann

The first DNS channel flow results at $\mathrm{Re}_\tau = 180$ were obtained by Kim et al. [73], with a follow-up paper [106] adding data for flows up to $\mathrm{Re}_\tau = 590$. The results of Moser et al. [106] are particularly useful, as flow statistics are publicly available on the Internet, making it easy to directly compare them with new results. A similar on-line database was published by Hoyas and Jiménez [58], who ran a DNS with $\alpha_x = 8\pi$, $\alpha_z = 3\pi$ at $\mathrm{Re}_\tau = 2003$. The simulation used a large domain of $6144 \times 633 \times 4608$, taking 6M CPU hours to complete.

Vreman and Kuerten [148] compared 8 results of $\mathrm{Re}_\tau = 180$ channel flow simulations obtained with finite difference and Fourier-Chebyshev spectral methods. The majority of the compared simulations, half of which were performed by Vreman and Kuerten

themselves, used a $4\pi H \times 2H \times 4\pi/3$ geometry, with a non-uniform grid in the wall-normal direction, matching that used by Kim et al. [73]. A good agreement between all analyzed results was observed, with deviations for streamwise velocity lower than 0.8%, pressure lower than 2%, and root mean square values of the fluctuating velocities and turbulent dissipation components lower than 5%. The agreement was considered to provide support for the conjecture about the uniqueness of the statistically stationary state of turbulent channel flow.

Lattice Boltzmann

A number of papers have been published on the problem of simulating turbulent channel flow using LB models. Lammers et al. [89] used the standard LBGK model with the D3Q19 lattice and a modification of the equilibrium function improving incompressibility. The geometry of the flow was $\alpha_x = 16$ and $\alpha_z = 1$, as compared to $\alpha_x = 4\pi$ and $\alpha_z = 4\pi/3$ for the reference DNS. For initial conditions, a velocity profile of the same form as our Eq. (6.7) (with slightly different constant values) was used. This profile was perturbed with streamwise- or spanwise- aligned vortices, and a random perturbation was also added to the pressure field. The paper states that the BBL condition was used for the walls, specifying it as $f_i(r, t+1) = f_{-i}(r, t)$. Note that this is slightly different than Eq. (2.46), which could be an accidental error in the text or an actual difference in the implementation where the value of the opposite distribution is taken prior to relaxation. With this approach, very good agreement with the reference DNS results was obtained. The simulation stability limit was determined as $\Delta_+ \leq 2.3$.

Spasov et al. [131] used the LBGK for a fully resolved $\mathrm{Re}_\tau = 180$ DNS simulation at $\Delta_+ \approx 1.4$, as well as ELBM for simulations at coarser grids corresponding to $\Delta_+ \approx 2.8$ and $\Delta_+ \approx 5.6$. All simulations used a $\alpha_x = 4$, $\alpha_z = 1$ geometry, the D3Q15 lattice, and the half-way bounce-back scheme for walls. The initial conditions were formed by a superposition of the analytic log-law approximation of the mean velocity profile and fluctuations from a spectral DNS. The ELBM runs were started from rescaled LBGK results and used the equilibrium and solver described in [25]. Both the ELBM and LBGK results agreed with the reference DNS data, but they cannot be considered statistically converged, as the profiles at the upper and lower wall are visibly different. The authors note that explicitly in the text. They also note that the used geometry based on the MFU is not long enough for the most distant points to be uncorrelated. The authors further state that at $\Delta_+ = 2.8$ only the ELBM simulation remained stable, and that at $\Delta_+ = 5.6$ the ELBM produced unphysical results (the velocity profile contained discontinuities).

Freitas et al. [37] performed $\mathrm{Re}_\tau = 200$ simulations using LBGK with D3Q19 and D3Q27, MRT, and cascaded LB (CLB). The geometry of the simulation domain corresponded to one MFU, and with $H = 67.5$ the resolution was set at $\Delta_+ \approx 2.96$. Initial conditions for

the LBGK simulations used a logarithmic profile and artificial, divergence-free velocity fluctuations for LBGK. The no-slip walls were implemented using a „half-way bounce-back" method [18]. Bouzidi et al. however describe an interpolation method different from Eq. (2.45), and Freitas et al. do not explicitly state the order of the interpolation anywhere in the text. The authors found the LBGK to be superior in terms of both the quality of solutions and complexity of the required initialization procedure. Between D3Q19 and D3Q27, the latter produced smaller deviation from the DNS results.

6.3.2 LBGK

In our approach, to simulate turbulent flow through a channel, we used the D3Q19 lattice, single precision calculations and the round-off minimizing model [128].

Initial conditions The simulation was started with a uniform density of 1 everywhere in the domain. The velocity field was initialized using the following analytical approximation of the average streamwise velocity profile:

$$u_+(y_+) = \begin{cases} \left(0.41^{-1} \ln y_+ + 5.5\right) & \text{for } y_+ > 11.45 \\ y_+ & \text{for } y_+ \leq 11.45 \end{cases} \tag{6.7}$$

merging the log-law with a linear profile in the viscous sublayer in a single continuous function. This profile was then perturbed with a random divergence-free field. The distributions were initialized using the LBGK equilibrium distribution corresponding to that velocity field.

To generate the random perturbance we took a vector field of uniform random variates in $(-1, 1)$ and smoothed it out by a convolution with a $\sigma = H/8$ Gaussian kernel. We then computed the curl of this field and rescaled it so that the maximum value corresponded to $u_{\max} = 0.05$. To avoid introducing unphysical artifacts close to the wall, the random perturbance was also rescaled with $u(y)/u_{\max}$.

We waited $2t_f$ iterations for any initial transients to subside and for the flow to become statistically stationary. After that, flow statistics were collected every 20 iterations for at least $57t_c$ time steps.

Estimation of u_τ In order to verify the Re_τ of the simulation as well as rescale all quantities of interest to the wall unit system it is necessary to accurately estimate u_τ. In a resolved simulation, this can be done directly from the definition of τ_w Eq. (6.3), either from the stress tensor obtained from the simulation, or from an estimate of the velocity derivative. Since in our simulations we did not save the stress tensor, we opted for the latter method. We took the first 10 points of the mean streamwise velocity profile, fit

them with a 3rd order spline, and used the spline's derivative at $y = 0$ to compute the velocity gradient at the wall.

When the simulation is highly under-resolved, this method is however no longer applicable. Indeed, since the viscous sublayer extends only up to $y_+ \approx 5$, and since the streamwise velocity profile is different in that sublayer and in the buffer region above it, we can easily end up with only one node (the wall) within the sublayer. In these situations, we used the original estimate of u_τ that was used as the initial condition for the simulation (Eq. (6.7) with $y_+ = \mathrm{Re}_\tau$).

Results We started by testing two channel geometries (α_x, α_z): (a) $(6, 2)$ and (b) $(12, 4)$ at a high resolution $H = 120\,(\Delta_+ = 1.5)$ and $\mathrm{Re}_\tau = 180$. As both geometries yielded results of similar quality, we used geometry (a) for the remaining simulations. We then gradually reduced the size of the lattice to test its impact on simulation precision and stability.

During the simulation, components of the Reynolds stress, as well as the first four powers of the components of the velocity field were numerically averaged in double precision directly on the GPU through periodic sampling. This made it possible to gather the necessary data without impacting the performance of the simulation.

The results remain of good quality until $H \approx 50$ (see Figure 6.5). At $H = 30$ the BBL results are no longer reliable, probably due to oscillations generated by the wall nodes (see Figure 6.3(b)). At $H = 35$ (BBL) and $H = 30$ (HBB) the results still provide a decent approximation of the fully resolved profiles.

In terms of numerical stability, both BBL and HBB displayed performance far exceeding previously reported results. With BBL the flow remained turbulent and stable $H = 30\,(\Delta_+ = 6)$, although the quality of the solution degraded visibly (standing oscillations in the flow field were visible). HBB performed even better, with flows stable at the same lattice sizes as BBL, and no anomalous patterns visible in the flow field (see Figure 6.3). At lower H the simulation eventually crashed for both boundary conditions.

We repeated our simulations also for $\mathrm{Re}_\tau = 590$, for which DNS data is available [106], reaching similar conclusions. To test the impact of the relaxation model, we repeated one of the lower resolution simulations at $H = 100$ with the ELBM model. Since the results are nearly identical to those of LBGK (see Figure 6.6), we conclude that the boundary condition plays the primary role in the accuracy of the simulation. The entropic relaxation would become important if we started experiencing numerical instabilities, which was not the case here.

Comparing the results from Figure 6.5 and Figure 6.7, we note a trend of the simulation providing good results up to $\Delta_+ \approx 3.6$, up to which the error curve is relatively flat and beyond which it starts to rise quickly. It is quite remarkable that this resolution is more than twice as coarse as that required for a DNS, even though no subgrid modeling

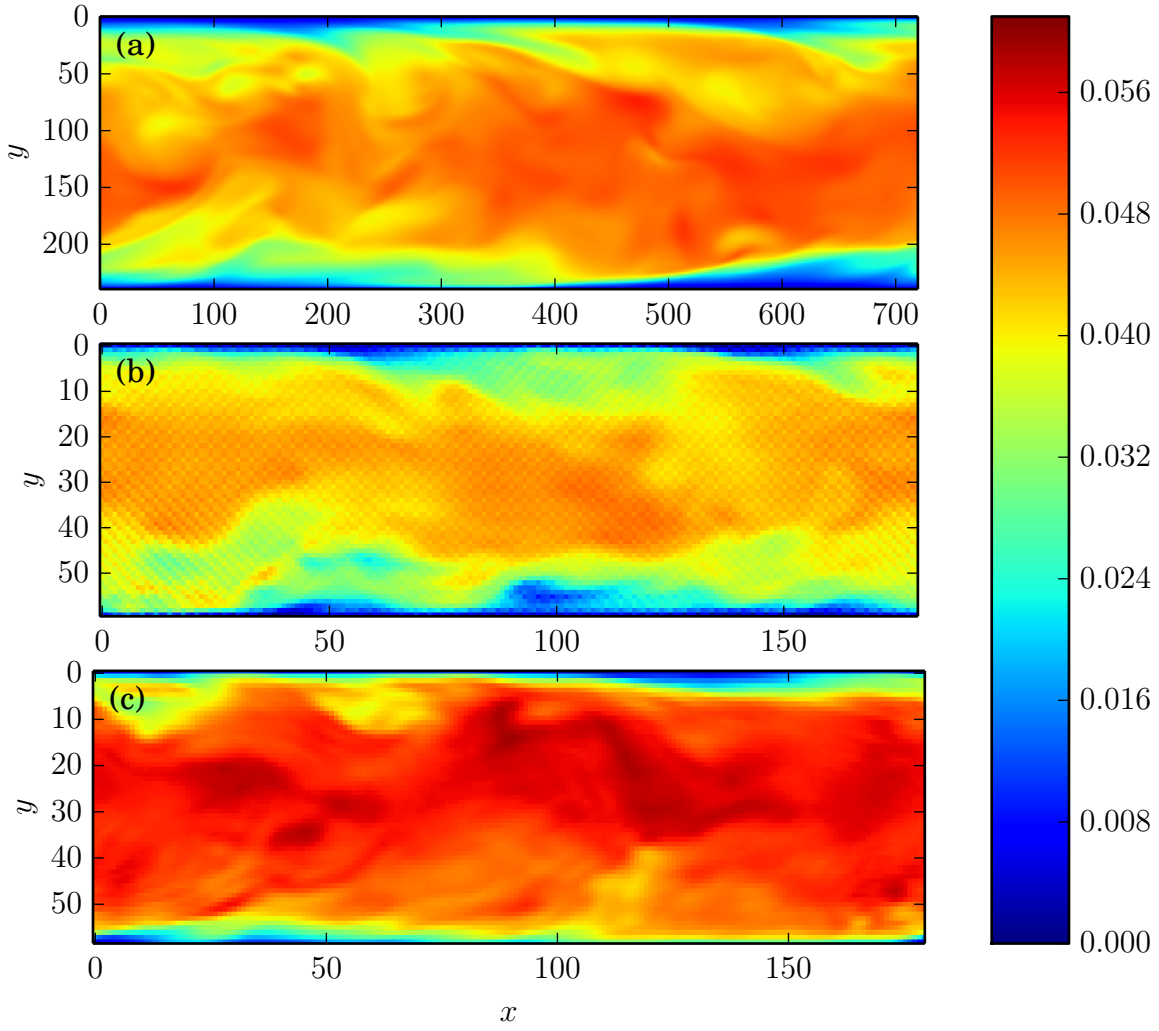


Figure 6.3: Streamwise velocity $u(z = 0)$ after $t = 3.5 \cdot 10^6$ iterations for (a) BBL, $H = 120$, (b) BBL, $H = 30$, (c) HBB, $H = 30$. Checkerboard pattern is visible with BBL when used at a very low resolution (panel (b)). Velocity and spatial coordinates are shown in lattice units.

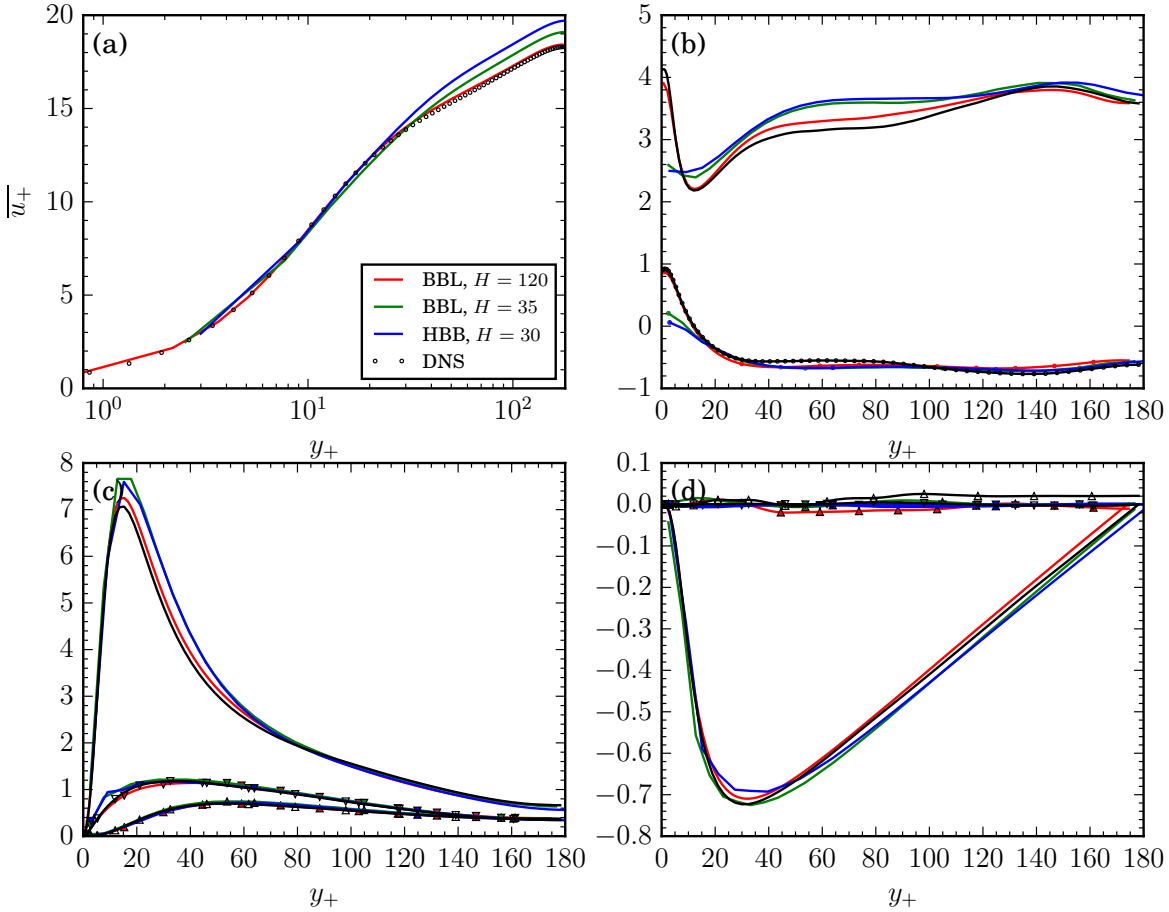


Figure 6.4: Profiles of (a) mean streamwise velocity $\overline{u}$, (b) skewness (smooth lines) and flatness (lines with dot markers), and (c-d) Reynolds stress tensor components: $\overline{u'^2}, \overline{u'v'}$ (smooth lines), $\overline{v'^2}, \overline{u'w'}$ ($\triangle$ lines), $\overline{w'^2}, \overline{v'w'}$ ($\triangledown$ lines). The DNS data [106] is for $\mathrm{Re}_\tau = 178.12$, while the LBM data is for $\mathrm{Re}_\tau = 174.43\,(H = 120)$, $176.59\,(H = 35)$, $179.69\,(H = 30)$ (based on the u_τ evaluation procedure described in the text).

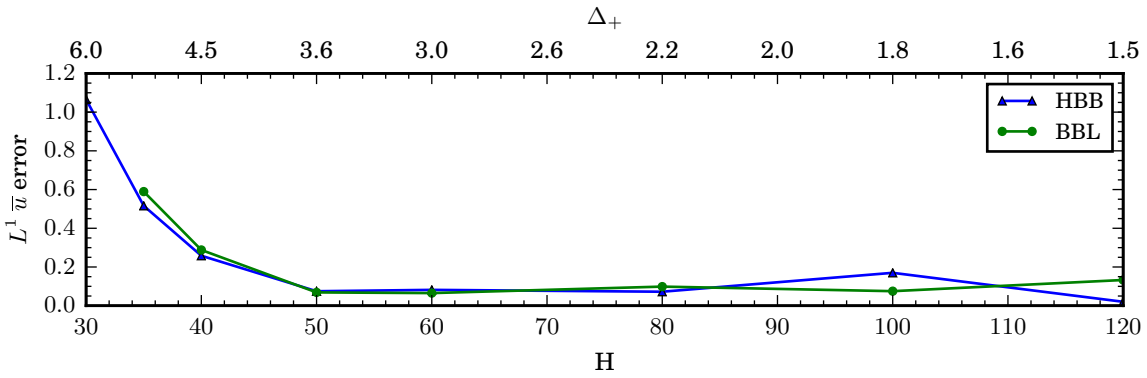


Figure 6.5: L^1 error of mean streamwise velocity profile as a function of lattice size at $\mathrm{Re}_\tau = 180$. The L^1 error is computed as $\sum_y |\overline{u}_{\mathrm{DNS}}(y) - \overline{u}_{\mathrm{LB}}(y)| / H$, where the reference DNS data [106] is resampled to locations corresponding to LB nodes through a third order spline fitted to the original data.

is attempted. The numerical value of the critical resolution Δ_+ also suggests that the main reason for the decreased quality of the results is insufficient resolution in the viscous sublayer.

6.3.3 Subgrid models

While most of the LBM studies of channel flows focused on the regime of $\mathrm{Re}_\tau \approx 200$, we decided to take advantage of the availability of spectral DNS data for channel flow at $\mathrm{Re}_\tau = 2000$ [58]. $\mathrm{Re}_\tau = 2000$ corresponds to $\mathrm{Re} \approx 9.6 \cdot 10^4$, and at such a high Reynolds number, only ELBM remained stable. We tested two domain configurations: $(2, 4, 12)$ at $H = 170$ ($\Delta_+ = 11.76$) and $(2, 2, 6)$ at $H = 270$ ($\Delta_+ = 7.41$), utilizing most of the memory available on 4 K40 GPUs (48 GB). The simulations ran at $u_0 = 0.025$, corresponding to $u_\tau = 1.04 \cdot 10^{-3}$ and used HBB for the walls. The $H = 270$ configuration was also tested with TMS walls.

We first visually inspected the results of the simulations (see Figure 6.8 for $H = 270$ and Figure 6.9 for $H = 170$) to verify that no unwanted featured (discontinuities, oscillations) were present, as reported in some of the prior work using ELBM [131]. No such features were observed. Figure 6.8(b,c) makes it possible to see how vortices are formed close to the wall and then propagate towards the center of the domain. We note that the flow field appears much more complex and chaotic than that previously seen in $\mathrm{Re}_\tau = 180$ simulations.

We also decided to inspect the flow field in the vicinity of the wall (see Figure 6.10), which revealed the previously discussed streak structure. Figure 6.10 shows data from

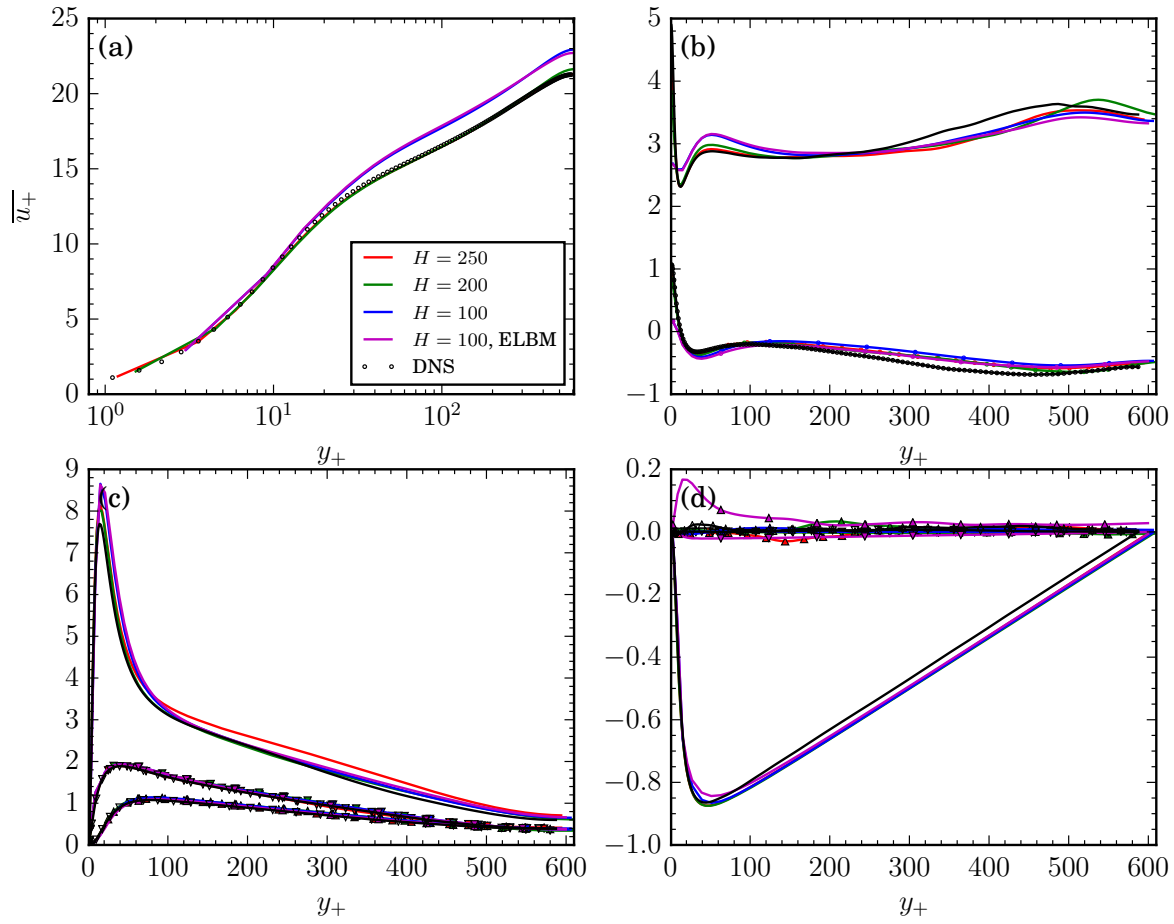


Figure 6.6: Profiles of (a) mean streamwise velocity $\overline{u}$, (b) skewness (smooth lines) and flatness (lines with dot markers), and (c-d) Reynolds stress tensor components: $\overline{u'^2}, \overline{u'v'}$ (smooth lines), $\overline{v'^2}, \overline{u'w'}$ ($\triangle$ lines), $\overline{w'^2}, \overline{v'w'}$ ($\triangledown$ lines). The DNS data [106] is for $\mathrm{Re}_\tau = 587.19$, while the LBM data is for $\mathrm{Re}_\tau = 593.82\,(H = 250)$, $607.36\,(H = 200)$, $577.82\,(H = 120)$, $605.67\,(H = 100)$.

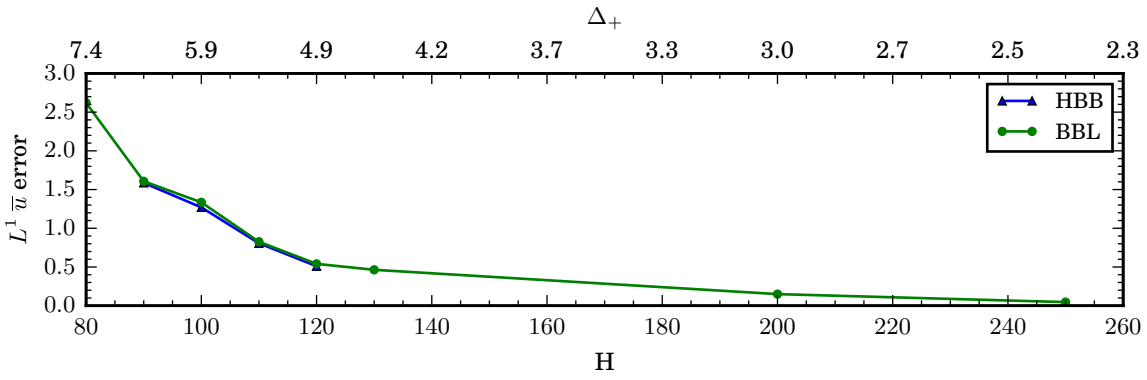


Figure 6.7: L^1 error of mean streamwise velocity profile as a function of lattice size at $\mathrm{Re}_\tau = 590$. See Figure 6.5 for the details of how the error was computed.

the first fluid node layer directly adjacent to the wall. For $H = 270$, this corresponds to a location where the streaks are expected to be most pronounced. For $H = 170$, the streaks are interspersed with areas of faster flow, which is to be expected, since the slice is 50% farther away from the wall than Figure 6.10(a). We note that the results are consistent with the experimental observations of mean spanwise streak separation of about 100 wall units.

Comparing the profiles of mean streamwise velocity against the DNS data [58] we note the overall good agreement. The $H = 270$ simulations underpredicted the maximum velocity by about 3%. All 3 simulations overpredicted the mean flow speed around $y_+ = 100$. The lowest resolution simulation at $H = 170$ unsurprisingly produced the worst profile in the region of $y_+ \in (10, 100)$. The TMS results appear to be closest to the DNS data, if the effective wall location for TMS is assumed to be the same as the wall location for BBL.

All simulations ran for $3.5 \cdot 10^6$ steps, which was dictated by the computational power available for our tests. We should note that if the first $1t_f$ iterations are discarded to allow for transients to disappear, this leaves only approximately $7t_c$ for averaging in the case of $H = 270$ and $9t_c$ in the case of $H = 170$. These times are not long enough to obtain good convergence. This was visible in the u profiles which were not ideally symmetrical (we noticed relative differences of up to 1.7%).

This problem is even more visible in Figure 6.12 and Figure 6.13 which show much more significant discrepancies. Without more data it is impossible to say how close they could get to the DNS in the limit of much longer simulation times. The data presented in these figures should therefore be only considered preliminary. We do however want to make two observations. First, the lowest resolution simulation at $H = 170$ showed discontinuities in $\overline{u'v'}$ and $\overline{v'}$ and $\overline{w'}$, which are likely caused by numerical instabilities very close to the wall, and as such unlikely to disappear with longer averaging times.

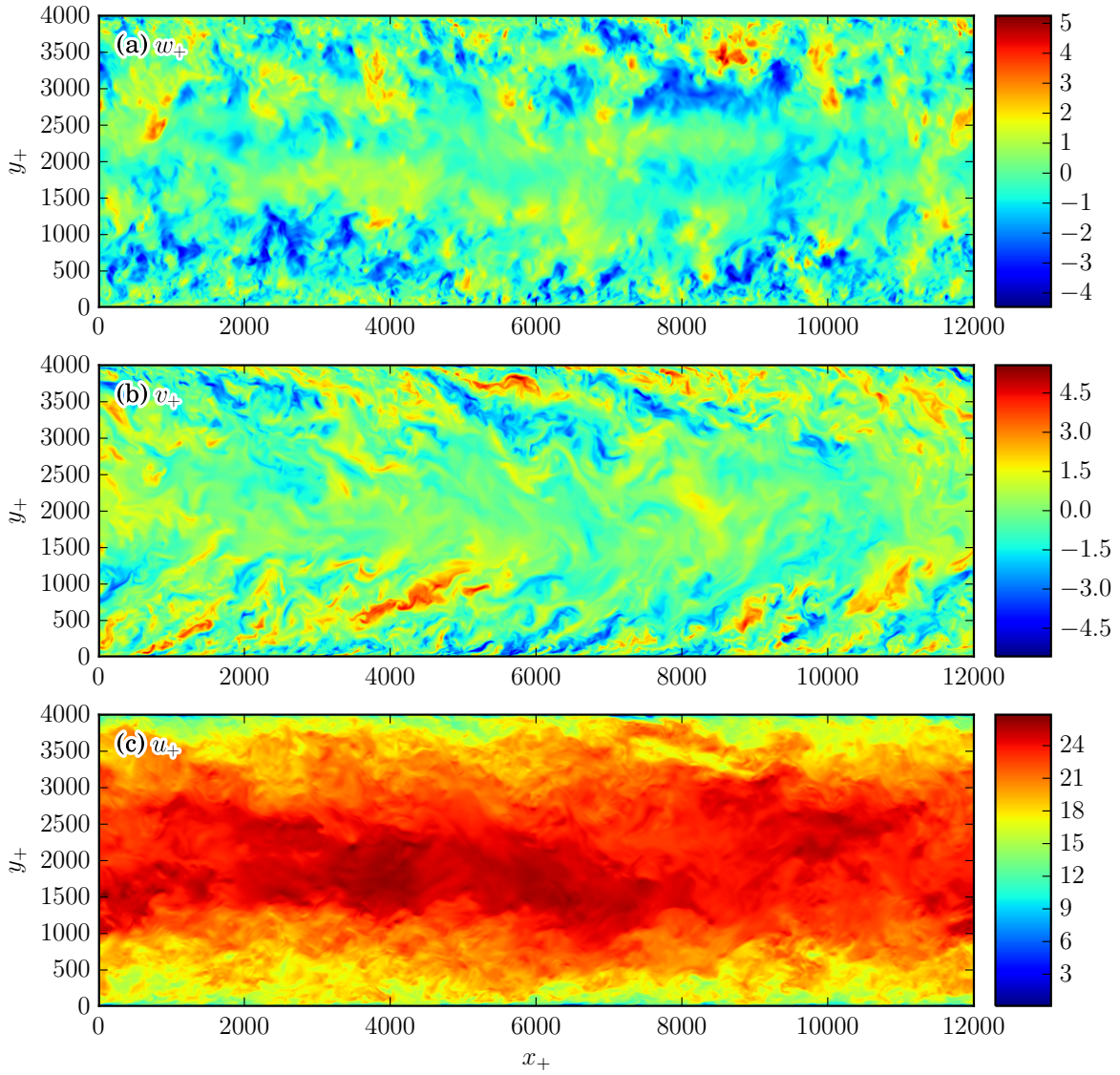


Figure 6.8: Instantaneous velocity field components (spanwise (a), wall-normal (b), streamwise (c)) in a $\mathrm{Re}_\tau = 2000$ flow at $H = 270$ in a plane parallel to the flow direction. Walls are located at $y_+ = 0$ and $y_+ = 4000$.

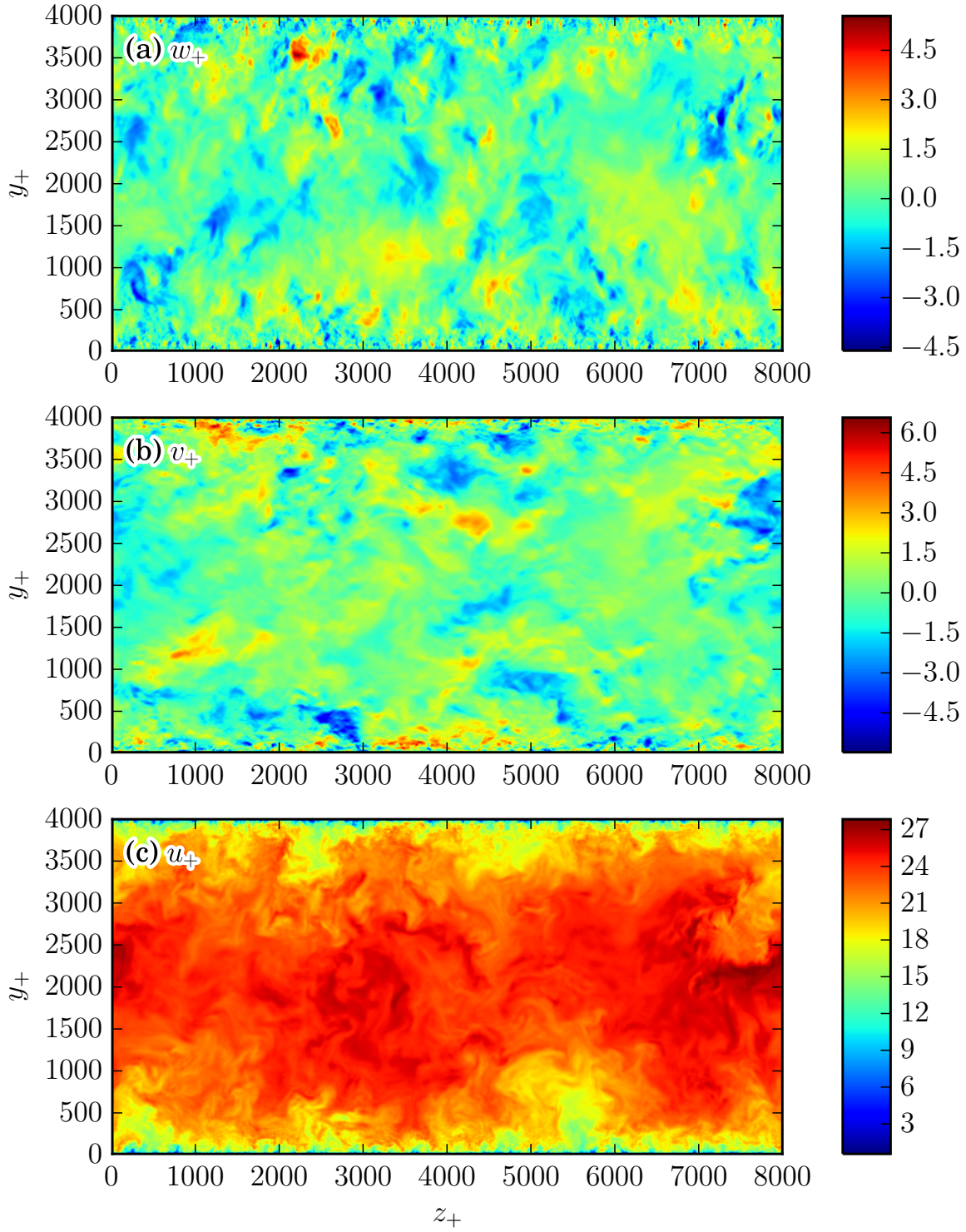


Figure 6.9: Instantaneous velocity field components (spanwise (a), wall-normal (b), streamwise (c)) in a $\mathrm{Re}_\tau = 2000$ flow at $H = 170$ in a plane orthogonal to the flow direction. Walls are located at $y_+ = 0$ and $y_+ = 4000$.

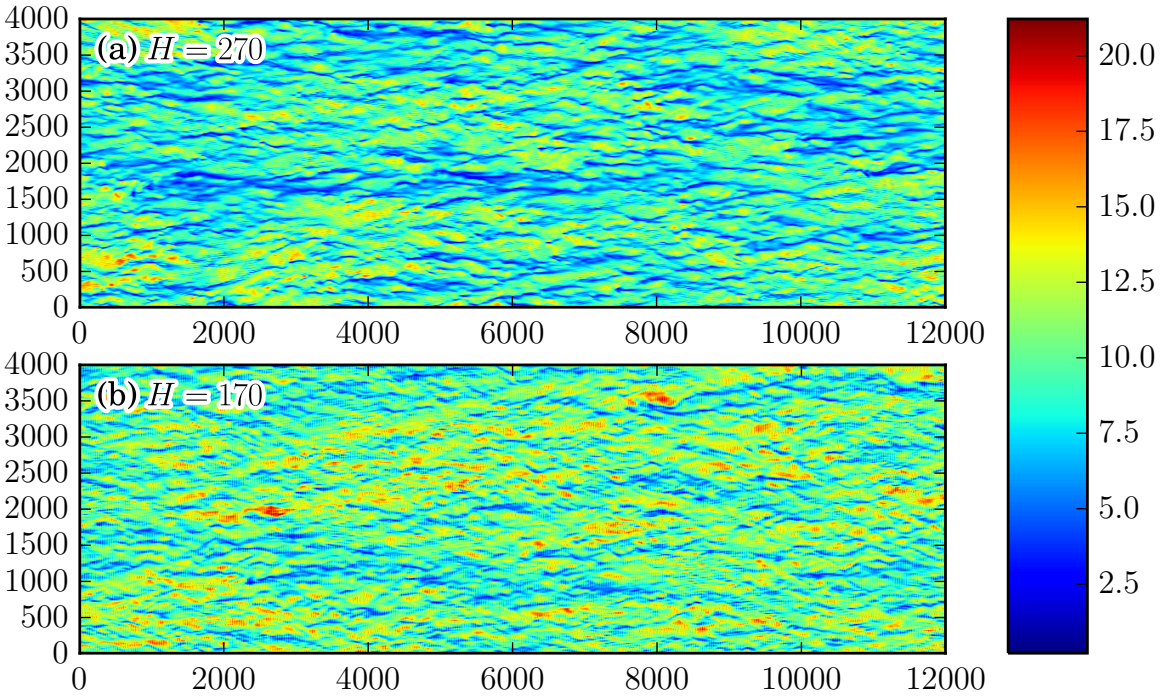


Figure 6.10: Streak structure in u_+ field close to the wall in $\mathrm{Re}_\tau = 2000$ simulations. Top panel: $H = 270, y_+ \approx 11.1$. Bottom panel: $H = 170, y_+ \approx 17.6$. Spatial coordinates are shown in wall units.

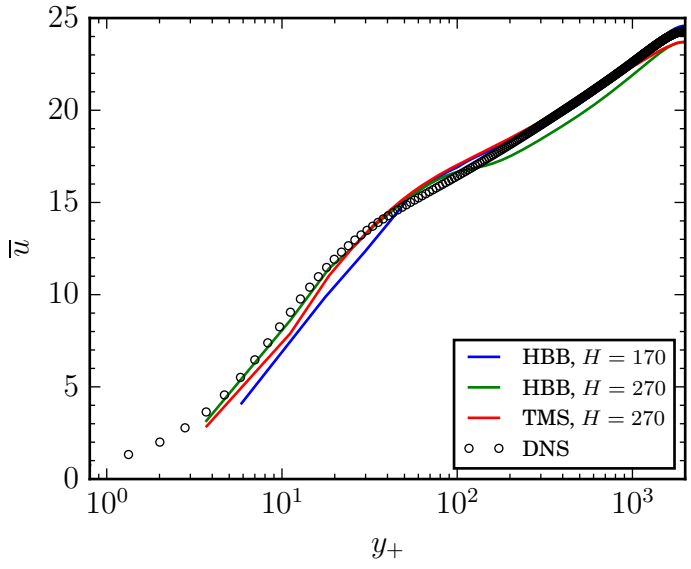


Figure 6.11: Profiles of mean streamwise velocity $\overline{u_+}$ in $\mathrm{Re}_\tau = 2000$ simulations with ELBM and HBB/TMS. DNS data comes from [58].

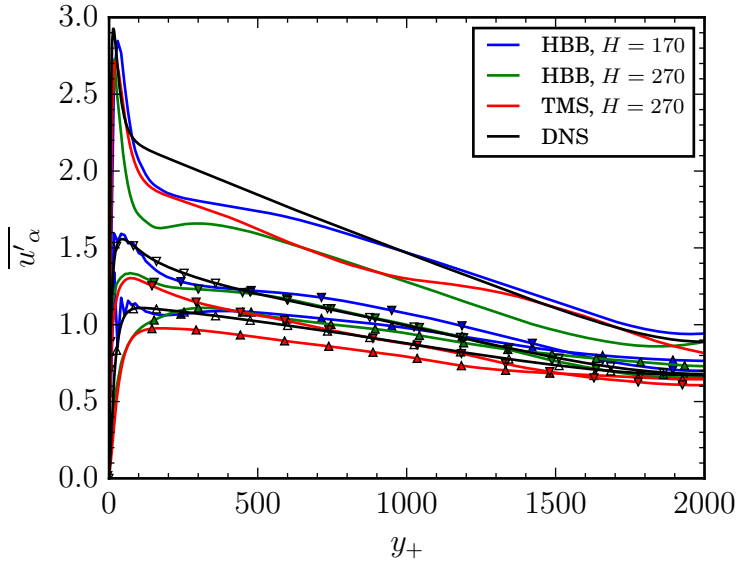


Figure 6.12: Profiles of velocity fluctuation intensities: $\overline{u'}$ (no markers), $\overline{v'}$ ($\triangle$), $\overline{w'}$ ($\bigtriangledown$) in $\mathrm{Re}_\tau = 2000$ simulations with ELBM and HBB/TMS. DNS data comes from [58].

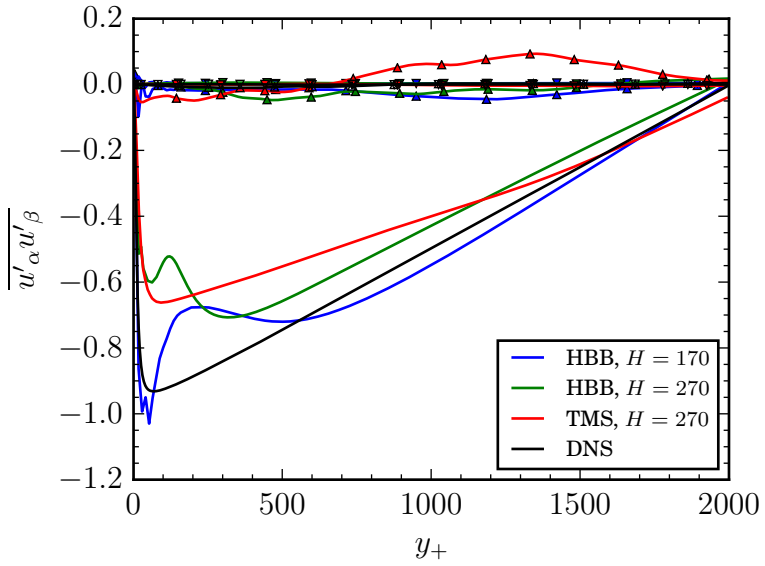


Figure 6.13: Profiles of Reynolds stress components: $\overline{u'v'}$ (no markers), $\overline{u'w'}$ ($\triangle$), $\overline{v'w'}$ ($\bigtriangledown$) in $\mathrm{Re}_\tau = 2000$ simulations with ELBM and HBB/TMS. DNS data comes from [58].

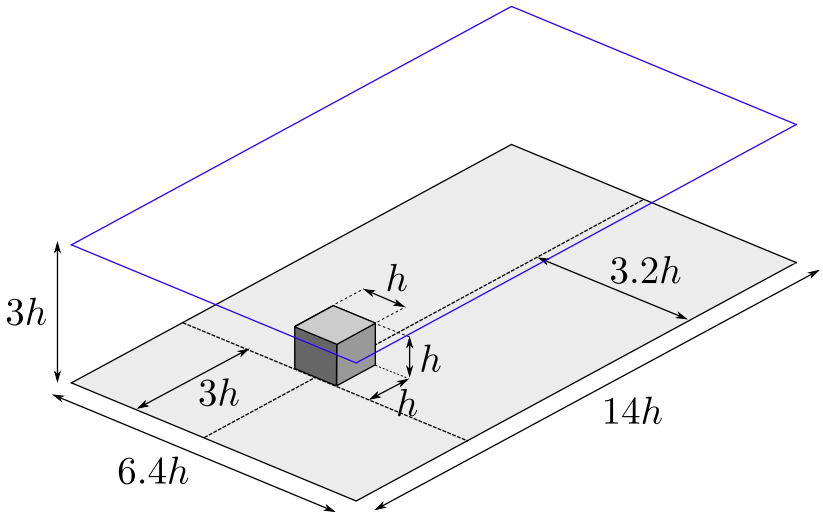


Figure 6.14: Geometry of the wall-mounted cube channel case.

Second, we note that TMS consistently provided smoothest profiles, despite averaging time and geometry identical to that of HBB.

6.4 Turbulent flow around a wall-mounted obstacle

To further test the behavior of bounce-back boundary conditions in turbulent flows, we also simulated the flow through a channel with a wall-mounted cube. This geometry is more demanding on the boundary conditions due to the presence of corners and edges in the cube, as well as surfaces orthogonal to the flow direction.

For our simulations we used the same geometry as that presented in [157], with which we compared our results. The size of the main channel was $14h \times 3h \times 6.4h$, with a cube of edge size h placed at the bottom wall, $3h$ from the channel entrance and spanwise in the middle of the computational domain (see Figure 6.14).

6.4.1 Prior work

The wall-mounted cube setup is a benchmark problem in the field of turbulent flows. The problem has been studied experimentally [95, 101–103] as well as through simulations, mainly using RANS [62] and LES [120, 121] turbulence models. The unsteady RANS and LES simulations showed good agreement with experimental data, reproducing global flow features.

Yakhot et al. [156, 157] performed a DNS simulation of this flow at $\mathrm{Re}_h = 1870$ using an immersed boundary method to implement the cuboid obstacle. Their simulation used a non-uniform mesh with $181 \times 121 \times 256$ grid points, with densest grid close to the channel and cube walls. Their work will serve us the main reference against which the results of the LBM simulation will be compared. We note that the geometry of the channel and the Re of the flow is slightly different than that of the experimental setup of [103], which used $\mathrm{Re}_h = 4440$, and $H/h = 3.3$.

6.4.2 Simulation setup

For the cube geometry, periodic boundary conditions can no longer be used in the streamwise direction to avoid coupling the flow at the entrance of the channel to the vortices generated by the obstacle, so following Yakhot et al., we added an additional *entrance* subdomain of size $9h \times 3h \times 6.4h$. The entrance subdomain used the standard channel geometry and periodic boundary conditions, and was completely independent of the rest of the simulation. Its only use was to generate a dynamic inflow profile for the main channel where the cube was located. This was achieved simply by performing the LB streaming step from the entrance to the main channel, but not vice versa. At the outflow of the channel, we used the equilibrium boundary condition with a constant density set to 1. The simulation was performed with D3Q19 LBGK relaxation model with half-way bounce-back walls, at $u_0 = 0.025$, $h = 40$ and $\mathrm{Re} = u_0 h/\nu \approx 2180$, corresponding to $\mathrm{Re}_\tau \approx 180$. The complete domain size was $980 \times 120 \times 256$ and the spatial resolution can be estimated at $\Delta_+ = 3$. In the discussion of the results, nondimensionalization was performed either using wall units assuming $u_\tau \approx 1.3762 \cdot 10^{-3}$ or with h for the spatial scale and a bulk velocity $u_b \approx 2.1663 \cdot 10^{-2}$ for velocity, as noted in the text.

6.4.3 Data analysis

A visual inspection of the instantaneous velocity field of the simulation (see Figure 6.15) allows us to determine that no unexpected features are present. In particular, the simulation appears to be free of numerical instabilities or unphysical oscillations that are sometimes triggered close to the walls. As could be expected, vortices are shed from the top and side of the cube, and a zone of backward flow develops directly behind it. More information about the flow structure is provided by the time-averaged velocity field, the streamlines of which we show in Figure 6.16. Several characteristic features are visible, similar to those reported by Meinders et al. [103]. A horseshoe vortex develops close to the wall, starting in front of the cube and enveloping it from the sides. An arc-type vortex is also visible around the cube, as well as a recirculation zone behind it.

A more quantitative picture is provided by Figure 6.17, which shows the streamlines in

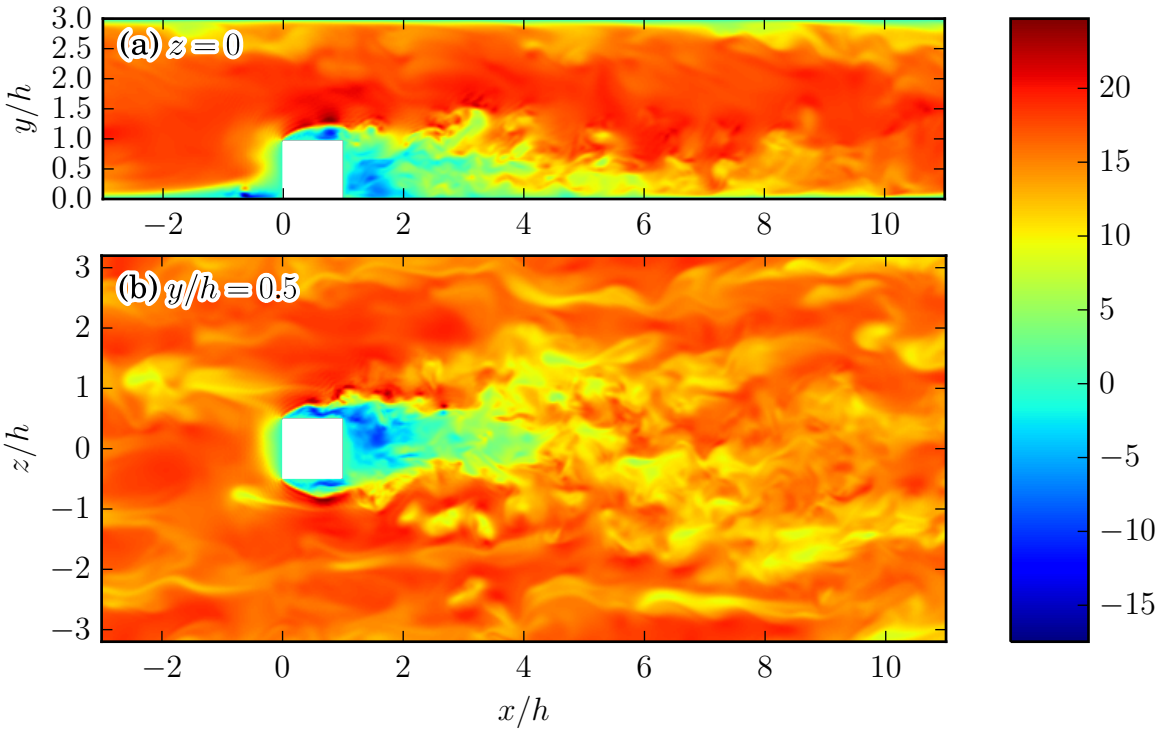


Figure 6.15: Instantaneous u_+ after $3.5 \cdot 10^6$ LB steps.

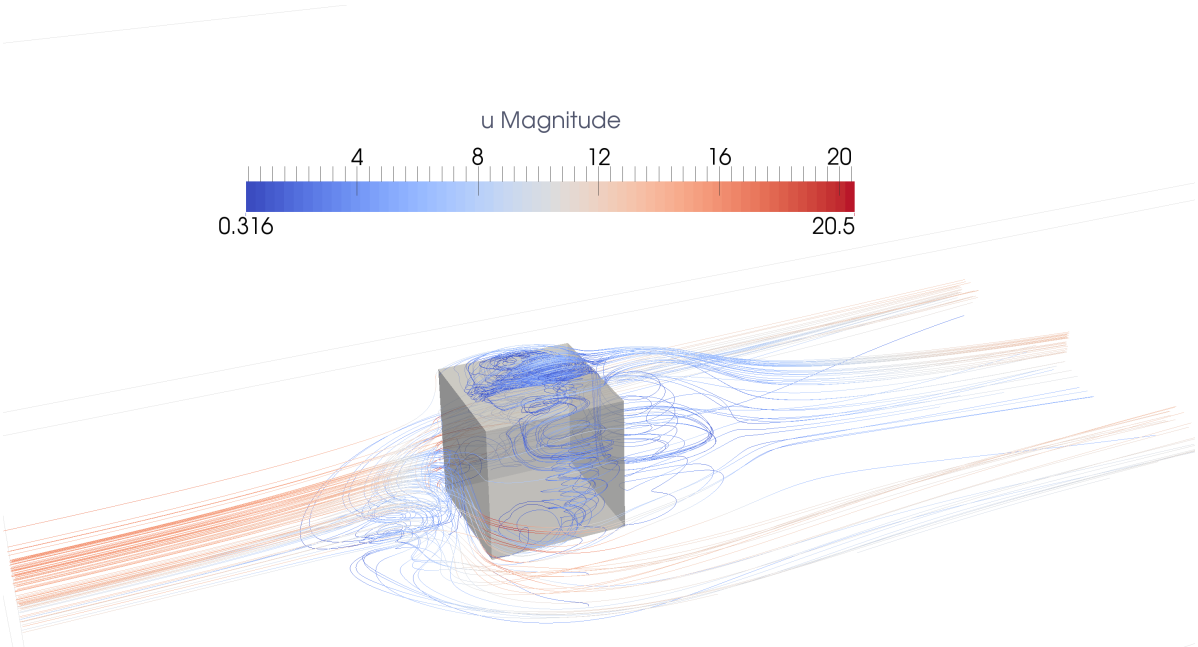


Figure 6.16: Time-averaged streamlines in turbulent flow around a wall-mounted cube. Velocity magnitude is shown in wall units.

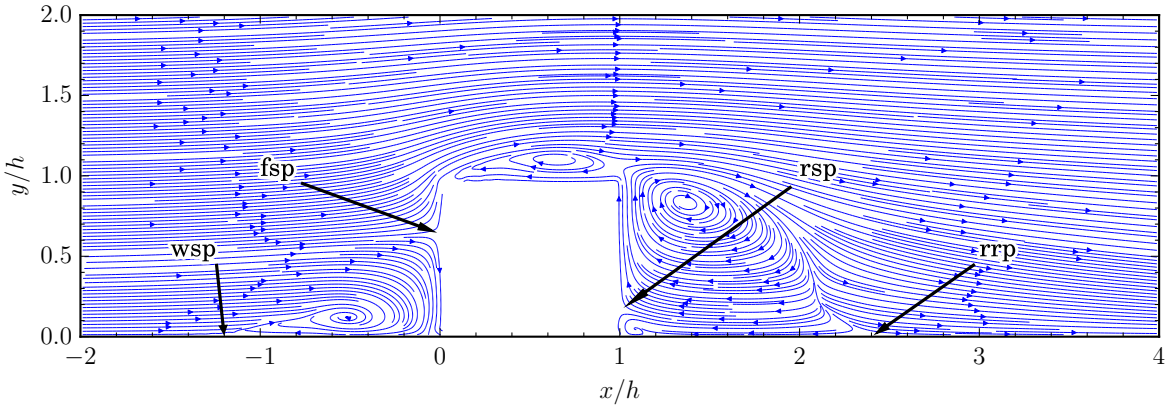


Figure 6.17: Time-averaged streamlines at $z = 0$ (symmetry plane). A number of flow features are annotated: windward separation point (wsp), front stagnation point (fsp), rear stagnation point (rsp), and rear reattachment point (rrp).

the symmetry plane. Three vortices are clearly visible — one in front of the cube (the horseshoe vortex), one on top of it (part of the arc vortex) and one behind it in the recirculation zone.

We analyzed the streamline plot in Figure 6.17, as well as its higher resolution versions (not shown here) in order to determine the spatial location of several characteristic points in the flow field. The distance between the windward separation point, where the vortex in front of the cube starts to form, and the front edge of the cube can be estimated to be $1.2h$ (Yakhot et al. [157] reported $1.21h$). The front stagnation point where the downward and upward flow separate, is $0.64h$ above the bottom wall (vs $0.65h$ in [157]). The rear stagnation point, fulfilling the same role behind the cube is at $0.16h$ (vs $0.15h$ in [157]). The rear reattachment point is $1.4h$ behind the cube (vs $1.5h$ in [157]). Finally, the center of the horseshoe vortex is slightly shifted towards the inlet ($(0.525h, 0.125h)$ in LBM vs $(0.45h, 0.12h)$ in [156]).

Overall, we can conclude that the location of the various flow features in the symmetry plane is in good agreement with the results reported in Yakhot et al. [157], especially when the relatively low lattice resolution used in our simulations is taken into account. One feature that we note is different than what Yakhot et al. reported is the lack of flow reattachment on top of the cube, which puts our results more in line with experimental data [103].

Streamlines in planes orthogonal to the symmetry plane are shown in Figure 6.18. Our results match those from Fig. 5 in [157]. We note that the imprint of the horseshoe vortex in front of the cube stops being visible already at $y = 0.25h$. A slight difference in the location of the side vortex can be seen between our data and that from Yakhot et al. [157], with the vortex being shifted closer to the rear of the cube in LBM data.

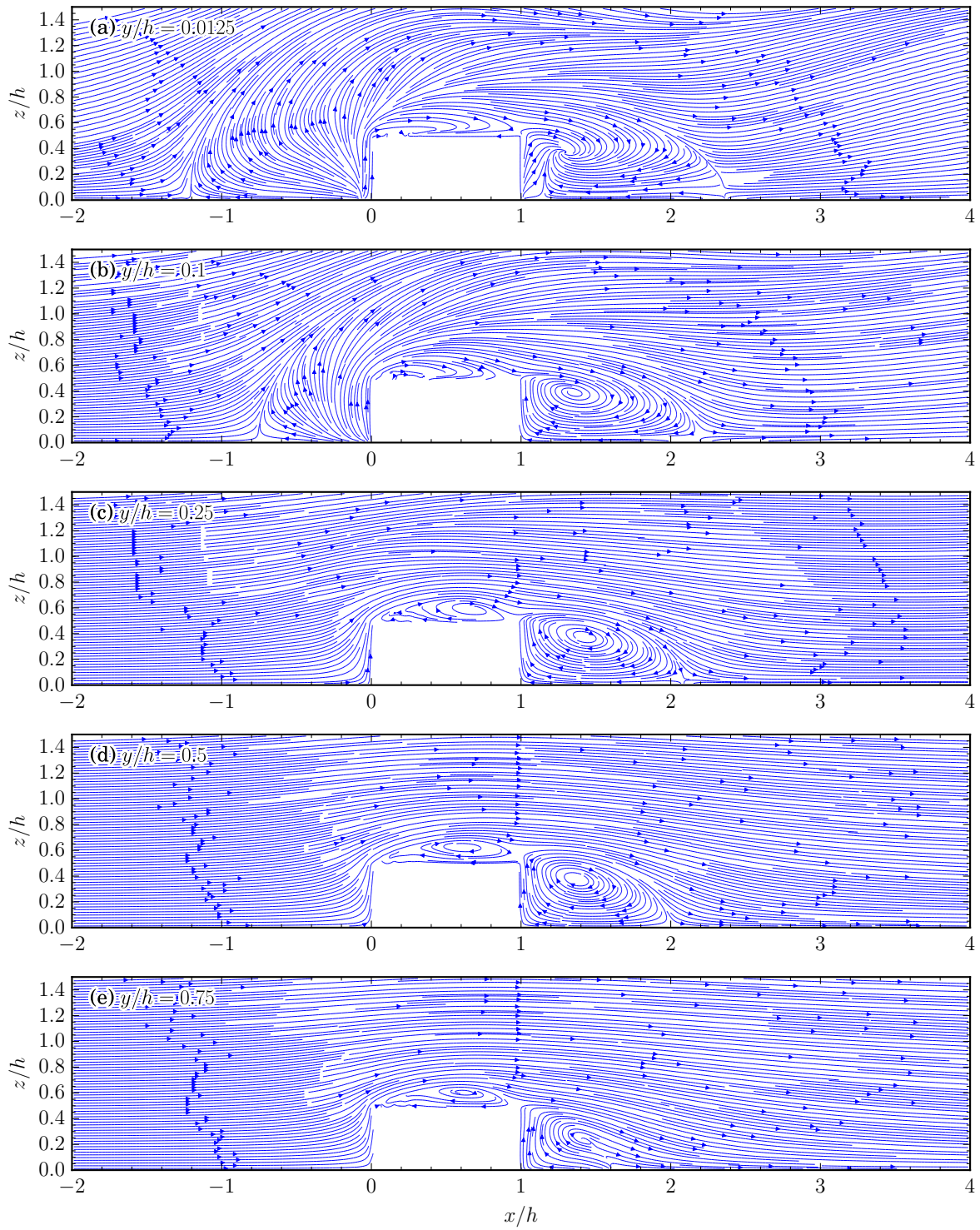


Figure 6.18: Time-averaged streamlines in the xz plane at $y/h = 0.0125$ (a), 0.1 (b), 0.25 (c), 0.5 (d), 0.75 (e), cf Fig. 5. in [157]. Only $z \geq 0$ is shown due to the symmetry of the problem.

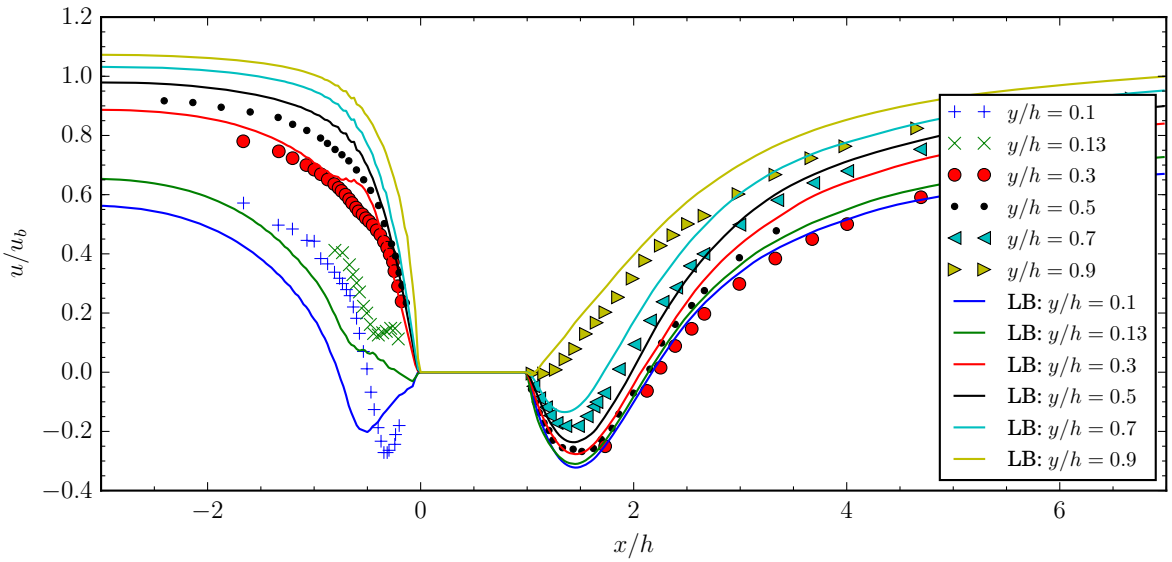


Figure 6.19: Time-averaged profiles of streamwise velocity u in the symmetry plane, at various values of y, cf. Fig. 9 in [157]. Experimental data points were extracted from Fig. 5 in [103] using Plot Digitizer.

This shift is in line with the lack of reattachment of the top vortex seen in Figure 6.17. The recirculation region in the wake behind the cube can be seen to retain approximately the same size in $y \in (0.1h, 0.5h)$, in line with the results of Yakhot et al. [157].

We do not show streamlines very close to the wall ($y/h = 0.003$ in [157]), as the uniform grid used in our simulation does not give us access to data at this location. Our near wall streamlines are shown in Figure 6.18(a), which is 4 times higher than the measurement location used in [157]. Nevertheless, a good agreement between the two simulations can be seen. We note the similar location of the saddle point where the flow separates in front of the cube. The rear vortex however, appears to be about 15% shorter in the LBM data.

Comparing the time-averaged profiles of streamwise velocity in the symmetry plane (see Figure 6.19) with Meinders et al. [103] and Yakhot et al. [157], we observe a match to the experimental data of comparable quality as that of Yakhot et al. [157]. These differences can likely be explained by the lower effective Re used in the numerical simulations, as well as some geometry differences between the simulations and the experiment (e.g. the channel height to cube height ratio was smaller than in the experiments). Comparing our data with Fig. 9 in [157] we note that our $u(y = 0.1h)$ profile shows a slightly deeper dip in front of the cube and has a lower value far away from the cube. The same is true, though to a lower degree, for $u(y = 0.3h)$ and $u(y = 0.5h)$, putting out $u(y = 0.3h)$ profile closer to the experimental data in front of the cube.

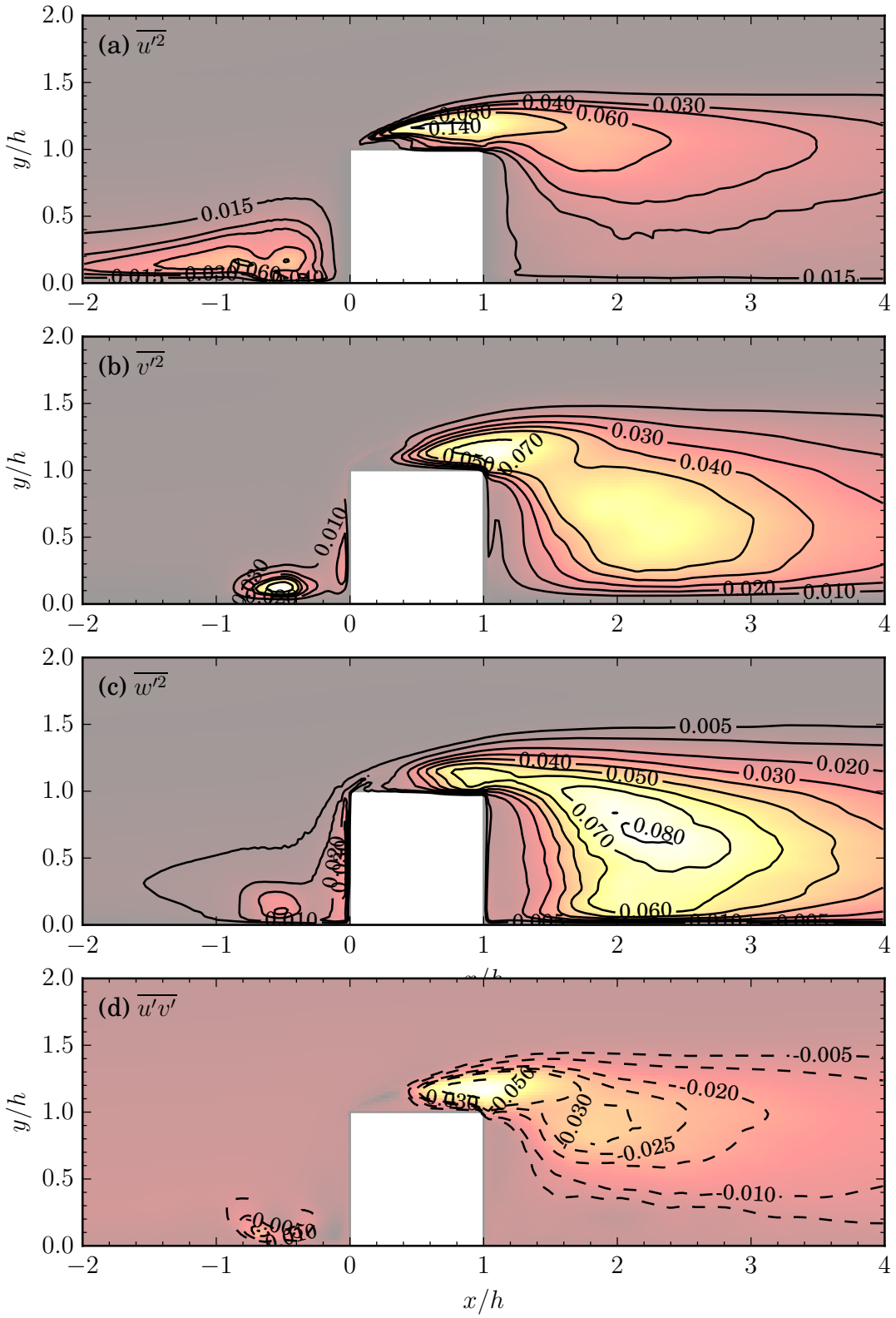


Figure 6.20: Mean-square intensities $\overline{u'^2}$ (a), $\overline{v'^2}$ (b), $\overline{w'^2}$ (c) and the Reynolds stress $\overline{u'v'}$ (d) in the symmetry plane $z = 0$, cf. Fig. 11 in [157]. Velocity is normalized by u_b.

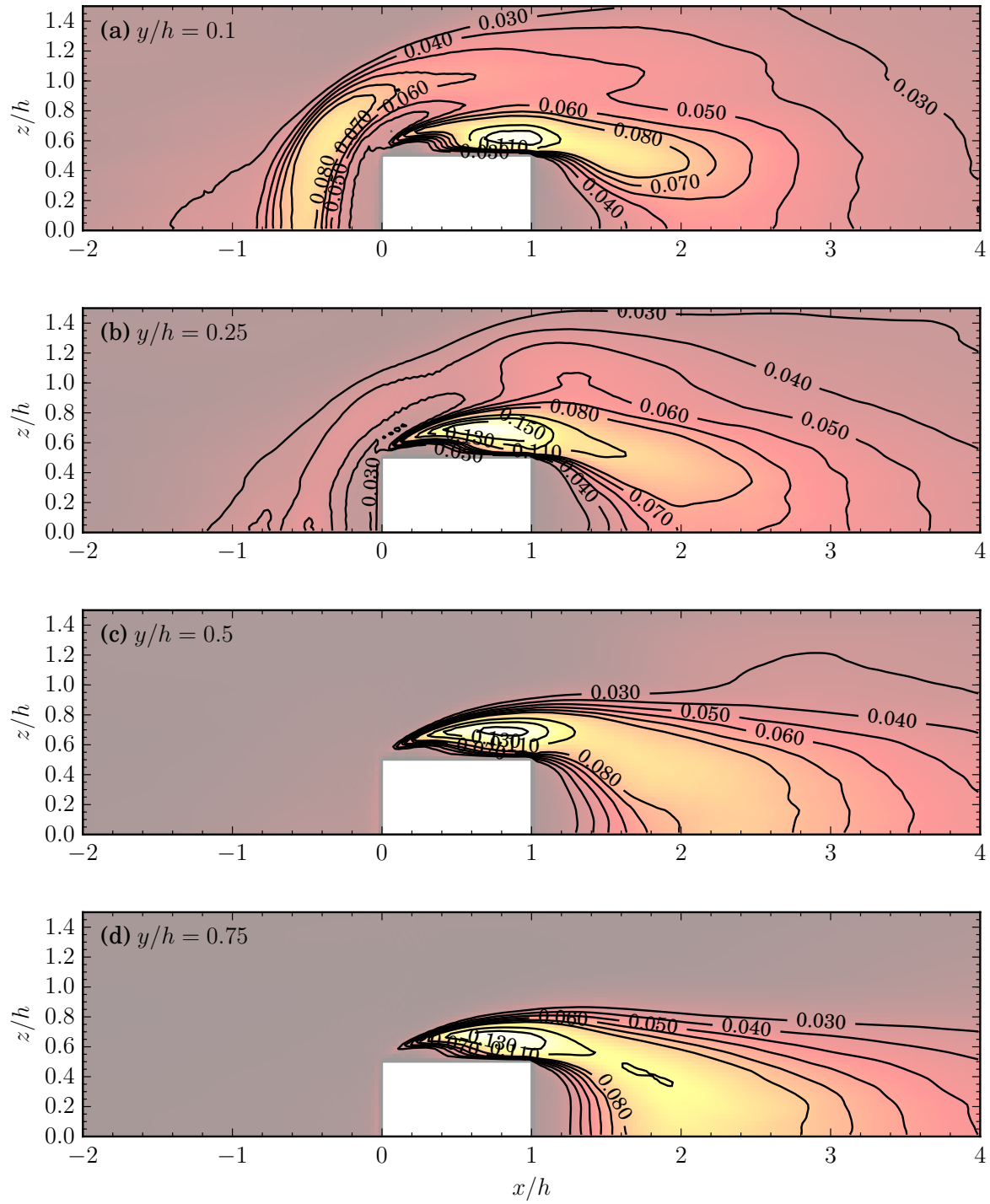


Figure 6.21: Turbulent kinetic energy $k = \left(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}\right)/2$ in the xz plane at $y/h = 0.1$ (a), $y/h = 0.25$ (b), $y/h = 0.5$ (c), $y/h = 0.75$ (d), cf. Fig. 12 in [157]. Velocity is normalized by u_b.

A comparison of higher-order statistics — the mean square intensities (Figure 6.20) and the turbulent kinetic energy (Figure 6.21) also shows good agreement with Yakhot et al. [157]. There is a general trend for the values from the LBM simulation to be slightly higher, for instance the isoline of $\overline{u'^2} = 0.03$ extends beyond $2.5h$ behind the cube, the area of $\overline{v'^2} = 0.05$ is larger and forms a single connected component behind the cube, and $\overline{w'^2} = 0.08$ appears behind the cube, but not directly over it. Given the overall similar structure of the field, we note that some of the discrepancies might be attributable to subtle differences in the normalization scheme. We also note a lack of a $\overline{w'^2} = 0.01$ structure connected to the rear face of the cube, which could be caused by the lower resolution of our simulation in this area. Similar conclusions hold for the Reynolds stress $\overline{u'v'}$ (Figure 6.20(d)), the distribution of which can be compared with Fig. 15 in [157].

Turbulent kinetic energy in Figure 6.21 is also in good agreement with the DNS results of Yakhot et al. [157]. We observe enhancement of turbulence production in the horseshoe vortex and the arc vortex, and note that the distribution of kinetic energy in the arc vortex is largely y-independent, in contrast to the horseshoe vortex.

7 Summary

In the previous chapters we have successfully demonstrated the applicability of the LBM in three different settings — bubble flow in microchannels, hemodynamical simulations in human blood vessels, and turbulent simulations at low to moderate Reynolds numbers. Our results showed good agreement with prior numerical and experimental work, and the proposed solution method was shown to be competitive with alternative solutions. The use of GPUs made it possible for most of the simulations to finish within several hours, with the exception of some of the turbulent flows where the necessity of long averaging times and large lattices meant that a full run took a few days to complete.

While the issue of speed might appear trivial, for practical reasons it is actually extremely important. Humans have limited attention spans, and research directions that take a long time to yield results might be explored late or not all. Even if the human factor is not considered to be an issue, computational power is always limited. Faster simulations thus directly translate into more research being done.

Overall, this work makes the following specific novel contributions:

- Provides an advanced, fully open source and free to use implementation of a wide range of lattice Boltzmann models for GPUs, validated on numerous test cases and comparing favorably with previously published results in terms of speed.
- Provides performance data over a wide range of relaxation models, lattices and GPU microarchitectures, aiding future simulation design.
- Shows how the entropic model can run at $\sim 75\%$ of the speed of LBGK, making it an attractive option for many simulations.
- Shows an effective implementation of multi-fluid models on GPUs.
- Demonstrates applicability of the free energy binary fluid model to Bretherton/-Taylor flows in microchannels and establishes resolution requirements for the film width.
- Provides a detailed comparison between GPU-based LBM and FVM in hemodynamical applications, taking into account both precision and speed in stationary and time-dependent flows, showing LBM to be up to 20 times more efficient.
- Compares three different methods of WSS estimation in LBM simulations of realistic, time-dependent hemodynamical flows.
- Shows stability limits in under-resolved turbulent channel DNS exceeding those

previously reported in the literature.

- Provides an evaluation of the ELBM in application to turbulent channel flow at $\mathrm{Re}_\tau = 2000$.
- Describes an LBM DNS of turbulent channel flow with a wall-mounted cube, including a detailed comparison with other DNS and experimental results.

We particularly hope that the Sailfish project will become a valuable research tool in various branches of science using computational fluid dynamics. A quick literature search reveals multiple research groups already using the code in their work [85, 97–99, 105, 115, 150, 160] — many of them even before the package was officially announced in a peer-reviewed publication [64]. The list above is likely incomplete, but we take it, together with an even larger number of works referring to but not using Sailfish directly, as an early indication of the usefulness of our work.

7.1 Future work

Our goal in this thesis was to show applicability of the LBM to a wide variety of physical systems. Time constraints limited the range and complexity of models that could be explored. In this section, we highlight a number of interesting potential extensions, identifying directions for follow-up work.

On the technical side, Sailfish could be extended with better handling of multi-GPU simulations on a single computational node. Currently, all inter-subdomain communication passes through the host. Modern GPUs however allow peer-to-peer GPU communication, which would make it possible to bypass the host completely. Also in case of multiple subdomains being handled by the same GPU, data between them should be transferred by direct memory reads.

It would be interesting to extend the Bretherton/Taylor flow simulations in 3D to other channel cross-sections — particularly ones that are not axis aligned. Channels with triangular and circular cross sections are used in practical applications, so being able to simulate them would be a valuable addition.

The hemodynamical simulations allow several interesting extensions that would enhance their applicability to other parts of the human circulatory system. For smaller blood vessels, blood could be modeled as a non-Newtonian, shear-thinning fluid relatively easily through local modification of the relaxation time, similar to one used in the Smagorinsky subgrid model. For even smaller vessels, where the discrete nature of the RBC suspension in blood plasma becomes important, RBCs could be modeled as solid bodies through an immersed-boundary scheme or through a modified multi-species scheme, similar to the Shan-Chen model. The former prospect is perhaps more interesting, as it could also be used to model distensible vessel walls, or other moving objects placed in the blood

stream, such as artificial heart valves.

A natural extension of the turbulent channel flow simulations would be to run them with a non-uniform lattice in the wall-normal direction — a practice employed by virtually all reported simulations using non-LB numerical methods. If LB simulations remain numerically stable despite the interpolation necessary between lattices of different densities, a significant reduction of computational cost could be achieved. It would also be interesting to test turbulent flows in more complex geometries, particularly ones where walls are not axis-aligned, and evaluate stability and accuracy of both simple bounce-back conditions as well as more complex solutions such the Bouzidi interpolated wall boundary condition or TMS. The Sailfish code could also be extended with on-GPU tracking of derivatives of the macroscopic fields, enabling analysis of quantities such as vorticity or the energy dissipation rate.

8 Acknowledgements

The development of the Sailfish code, as well as the work presented in this thesis would not have been possible without the support of various people and organizations, which I wish to gratefully acknowledge here.

I would like to thank the University of Silesia for hosting the website of the Sailfish project, Institute of Physics at the University of Silesia for hosting the GPU workstations on which a lot of the work presented here was done, NVIDIA Corp. for providing hardware and computational time for LB code development and numerical experiments, Vratis Ltd. for access to hardware for benchmarking the hemodynamics simulations and for access to their commercial GPU library for OpenFOAM, and the PLGRID project and ETH Zurich for access to GPU clusters.

I also acknowledge the support of the TWING project co-financed by the European Social Fund, two grants as part of the Młodzi Badacze programme at the Institute of Physics at the University of Silesia, and the computational grant „channelelbm" within the PLGRID project.

I am also particularly grateful for the help and support of the following people: Alexander Kuzmin, for making me interested in Bretherton flow problems, for his help with cross-validating early versions of the Sailfish code as well as helpful discussions and help with data analysis; Jakub Poła, for preparing parts of the geometries from the Aneurisk database and for performing FVM simulations; Ilya Karlin and Shyam Chikatamarla, for introducing me to the world of turbulent flow simulations, for sharing data from their Kida vortex simulations and helpful discussions about turbulent flows; Jerzy Łuczka and Marcin Kostur for their support during the time of my PhD studies and support of the Sailfish project; and least but not least, my parents and Natalia Woźnica for their unwavering encouragement.

Finally, I would like to thank all those I might have omitted above, particularly everyone who used Sailfish in their research, contributed bug reports, code changes or contributed to the discussions on the project mailing list.

Bibliography

[1] Aneurisk-Team. AneuriskWeb project website, http://ecm2.mathcs.emory.edu/aneuriskweb. Web Site, 2012. URL `http://ecm2.mathcs.emory.edu/aneuriskweb`.

[2] Ansumali, S., Karlin, I. V., and Öttinger, H. C. Minimal entropic kinetic models for hydrodynamics. *EPL (Europhysics Letters)*, 63(6):798, 2007. URL `http://iopscience.iop.org/0295-5075/63/6/798`.

[3] Artoli, A., Hoekstra, A., and Sloot, P. 3d pulsatile flow with the lattice boltzmann bgk method. *International Journal of Modern Physics C*, 13(08):1119–1134, 2002.

[4] Artoli, A., Hoekstra, A. G., and Sloot, P. M. Simulation of a systolic cycle in a realistic artery with the lattice boltzmann bgk method. *International Journal of Modern Physics B*, 17(01n02):95–98, 2003.

[5] Artoli, A., Kandhai, D., Hoefsloot, H. C., Hoekstra, A. G., and Sloot, P. M. Lattice bgk simulations of flow in a symmetric bifurcation. *Future Generation Computer Systems*, 20(6):909–916, 2004.

[6] Artoli, A. M., Hoekstra, A. G., and Sloot, P. M. Mesoscopic simulations of systolic flow in the human abdominal aorta. *Journal of biomechanics*, 39(5):873–884, 2006.

[7] Artoli, A. M., Hoekstra, A. G., and Sloot, P. M. Optimizing lattice boltzmann simulations for unsteady flows. *Computers & fluids*, 35(2):227–240, 2006.

[8] Axner, L., Hoekstra, A. G., and Sloot, P. M. Simulating time harmonic flows with the lattice boltzmann method. *Physical Review E*, 75(3):036709, 2007.

[9] Axner, L., Hoekstra, A. G., Jeays, A., Lawford, P., Hose, R., and Sloot, P. Simulations of time harmonic blood flow in the mesenteric artery: comparing finite element and lattice boltzmann methods. *Biomedical engineering online*, 8(1):23, 2009.

[10] Baieth, H. A. Physical parameters of blood as a non-newtonian fluid. *International journal of biomedical science: IJBS*, 4(4):323, 2008.

[11] Bailey, P., Myre, J., Walsh, S. D. C., Lilja, D. J., and Saar, M. O. Accelerating lattice boltzmann fluid flow simulations using graphics processors. In *Parallel Processing, 2009. ICPP'09. International Conference on*, page 550–557, 2009.

[12] Benzi, R. and Frisch, U. Turbulence. 5(3):3439, 2010.

[13] Berčič, G. and Pintar, A. The role of gas bubbles and liquid slug lengths on mass transport in the taylor flow through capillaries. *Chemical Engineering Science*, 52 (21):3709–3719, 1997.

[14] Berger, S. and Jou, L. Flows in stenotic vessels. *Annual Review of Fluid Mechanics*, 32(1):347–382, 2000.

[15] Bernaschi, M., Rossi, L., Benzi, R., Sbragaglia, M., and Succi, S. Graphics processing unit implementation of lattice boltzmann models for flowing soft systems. *Phys. Rev. E*, 80(6):066707, Dec 2009. doi: 10.1103/PhysRevE.80.066707.

[16] Bhatnagar, P. L., Gross, E. P., and Krook, M. A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems. *Physical Review Online Archive (Prola)*, 94(3):511–525, May 1954. doi: 10.1103/PhysRev.94.511. URL `http://dx.doi.org/10.1103/PhysRev.94.511`.

[17] Bondi, A. and Eirich, F. Rheology. theory and applications, vol. 3, 1960.

[18] Bouzidi, M., Firdaouss, M., and Lallemand, P. Momentum transfer of a boltzmann-lattice fluid with boundaries. *Physics of Fluids*, 13:3452, 2001.

[19] Bretherton, F. The motion of long bubbles in tubes. *Journal of Fluid Mechanics*, 10(02):166–188, 1961.

[20] Campbell, I. C., Ries, J., Dhawan, S. S., Quyyumi, A. A., Taylor, W. R., and Oshinski, J. N. Effect of inlet velocity profiles on patient-specific computational fluid dynamics simulations of the carotid bifurcation. *Journal of biomechanical engineering*, 134(5):051001, 2012.

[21] Carver, H. B., Nash, R. W., Bernabeu, M. O., Hetherington, J., Groen, D., Krüger, T., and Coveney, P. V. Choice of boundary condition and collision operator for lattice-boltzmann simulation of moderate reynolds number flow in complex domains. *arXiv preprint arXiv:1211.0205*, 2012.

[22] Chatzizisis, Y. S., Jonas, M., Coskun, A. U., Beigel, R., Stone, B. V., Maynard, C., Gerrity, R. G., Daley, W., Rogers, C., Edelman, E. R., et al. Prediction of the localization of high-risk coronary atherosclerotic plaques on the basis of low endothelial shear stress an intravascular ultrasound and histopathology natural history study. *Circulation*, 117(8):993–1002, 2008.

[23] Chen, S. and Doolen, G. D. Lattice boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30(1):329–364, 1998. doi: 10.1146/annurev.fluid.30.1.329. URL `http://dx.doi.org/10.1146/annurev.fluid.30.1.329`.

[24] Chikatamarla, S. and Karlin, I. Entropic lattice boltzmann method for turbulent flow simulations: Boundary conditions. *Physica A: Statistical Mechanics and its Applications*, Jan. 2013. ISSN 03784371. doi: 10.1016/j.physa.2012.12.034. URL `http://linkinghub.elsevier.com/retrieve/pii/S0378437113000113`.

[25] Chikatamarla, S., Ansumali, S., and Karlin, I. Entropic lattice boltzmann models for hydrodynamics in three dimensions. *Physical review letters*, 97(1):010201, 2006.

[26] Chikatamarla, S., Ansumali, S., and Karlin, I. Entropic lattice boltzmann models for hydrodynamics in three dimensions. *Physical Review Letters*, 97, July 2006. ISSN 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.97.010201. URL `http://link.aps.org/doi/10.1103/PhysRevLett.97.010201`.

[27] Chikatamarla, S. S., Frouzakis, C. E., Karlin, I. V., Tomboulides, A. G., and Boulouchos, K. B. Lattice boltzmann method for direct numerical simulation of turbulent flows. *Journal of Fluid Mechanics*, 656: 298–308. ISSN 1469-7645. doi: 10.1017/S0022112010002740. URL `http://dx.doi.org/10.1017/S0022112010002740`.

[28] Chiu, J.-J. and Chien, S. Effects of disturbed flow on vascular endothelium: pathophysiological basis and clinical perspectives. *Physiological reviews*, 91(1):327–387, 2011.

[29] Dai, G., Kaazempur-Mofrad, M. R., Natarajan, S., Zhang, Y., Vaughn, S., Blackman, B. R., Kamm, R. D., García-Cardeña, G., and Gimbrone, M. A. Distinct endothelial phenotypes evoked by arterial waveforms derived from atherosclerosis-susceptible and-resistant regions of human vasculature. *Proceedings of the National Academy of Sciences of the United States of America*, 101(41):14871–14876, 2004.

[30] Dellar, P. Incompressible limits of lattice Boltzmann equations using multiple relaxation times. *Journal of Computational Physics*, 190(2):351–370, Sept. 2003. ISSN 00219991. doi: 10.1016/s0021-9991(03)00279-1. URL `http://dx.doi.org/10.1016/s0021-9991(03)00279-1`.

[31] d'Humières, D., Bouzidi, M., and Lallemand, P. Thirteen-velocity three-dimensional lattice boltzmann model. *Phys. Rev. E*, 63: 066702, May 2001. doi: 10.1103/PhysRevE.63.066702. URL `http://link.aps.org/doi/10.1103/PhysRevE.63.066702`.

[32] d'Humières, D., Ginzburg, I., Krafczyk, M., Lallemand, P., and Luo, L.-S. Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Phil. Trans. R. Soc. Lond. A*, 360:437–451, 2002.

[33] Epstein, F. H., Gibbons, G. H., and Dzau, V. J. The emerging concept of vascular remodeling. *New England Journal of Medicine*, 330(20):1431–1438, 1994.

[34] Ferguson, G. G. Physical factors in the initiation, growth, and rupture of human intracranial saccular aneurysms. *Journal of neurosurgery*, 37(6):666–677, 1972.

[35] Fischer, P. F., Loth, F., Lee, S. E., Lee, S.-W., Smith, D. S., and Bassiouny, H. S. Simulation of high-reynolds number vascular flows. *Computer methods in applied mechanics and engineering*, 196(31):3049–3060, 2007.

[36] Ford, M. D., Alperin, N., Lee, S. H., Holdsworth, D. W., and Steinman, D. A. Characterization of volumetric flow rate waveforms in the normal internal carotid and vertebral arteries. *Physiological Measurement*, 26(4):477, 2005. URL `http://stacks.iop.org/0967-3334/26/i=4/a=013`.

[37] Freitas, R. K., Henze, A., Meinke, M., and Schröder, W. Analysis of lattice-boltzmann methods for internal flows. *Computers & Fluids*, 47(1):115–121, 2011.

[38] Fukagata, K., Kasagi, N., Ua-arayaporn, P., and Himeno, T. Numerical simulation of gas–liquid two-phase flow and convective heat transfer in a micro tube. *International Journal of Heat and Fluid Flow*, 28(1):72–82, 2007.

[39] Fung, Y.-C. *Biomechanics: mechanical properties of living tissues*. Springer Science & Business Media, 2013.

[40] Geerdink, J. B. and Hoekstra, A. G. Comparing entropic and multiple relaxation times lattice boltzmann methods for blood flow simulations. *International Journal of Modern Physics C*, 20(05):721–733, 2009.

[41] Giavedoni, M. D. and Saita, F. A. The axisymmetric and plane cases of a gas phase steadily displacing a newtonian liquid—a simultaneous solution of the governing equations. *Physics of Fluids (1994-present)*, 9(8):2420–2428, 1997.

[42] Guennebaud, G., Jacob, B., et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[43] Gunstensen, A. K., Rothman, D. H., Zaleski, S., and Zanetti, G. Lattice boltzmann model of immiscible fluids. *Phys. Rev. A*, 43:4320–4327, Apr 1991. doi: 10.1103/PhysRevA.43.4320. URL `http://link.aps.org/doi/10.1103/PhysRevA.43.4320`.

[44] Guo, Z., Zheng, C., and Shi, B. Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Phys. Rev. E*, 65(046308):1–6, 2002.

[45] Gupta, R., Fletcher, D., and Haynes, B. Taylor flow in microchannels: a review of experimental and computational work. *The Journal of Computational Multiphase Flows*, 2(1):1–32, 2010.

[46] Habich, J., Feichtinger, C., Köstler, H., Hager, G., and Wellein, G. Performance engineering for the lattice boltzmann method on gpgpus: Architectural requirements and performance results. *Computers & Fluids*, (0): –, 2012. ISSN 0045-7930. doi: 10.1016/j.compfluid.2012.02.013. URL `http://www.sciencedirect.com/science/article/pii/S0045793012000679`.

[47] Han, Y. and Shikazono, N. Measurement of liquid film thickness in micro square channel. *International Journal of Multiphase Flow*, 35(10):896–903, 2009.

[48] Hazel, A. L. and Heil, M. The steady propagation of a semi-infinite bubble into a tube of elliptical or rectangular cross-section. *Journal of Fluid Mechanics*, 470: 91–114, 2002.

[49] He, X. and Luo, L.-S. Theory of the lattice boltzmann method: From the boltzmann equation to the lattice boltzmann equation. *Phys. Rev. E*, 56:6811–6817, Dec 1997. doi: 10.1103/PhysRevE.56.6811. URL `http://link.aps.org/doi/10.1103/PhysRevE.56.6811`.

[50] He, X. and Luo, L.-S. Lattice boltzmann model for the incompressible navier–stokes equation. *Journal of Statistical Physics*, 88(3-4):927–944, 1997.

[51] He, X., Zou, Q., Luo, L.-S., and Dembo, M. Analytic solutions of simple flows and analysis of nonslip boundary conditions for the lattice boltzmann bgk model. *Journal of Statistical Physics*, 87(1-2):115–136, 1997.

[52] He, X., Duckwiler, G., and Valentino, D. J. Lattice boltzmann simulation of cerebral artery hemodynamics. *Computers & Fluids*, 38(4):789–796, 2009.

[53] Hecht, F. New development in freefem++. *J. Numer. Math.*, 20(3-4):251–265, 2012. ISSN 1570-2820.

[54] Heil, M. Finite reynolds number effects in the bretherton problem. *Physics of Fluids (1994-present)*, 13(9):2517–2521, 2001.

[55] Hirabayashi, M., Ohta, M., Rüfenacht, D. A., and Chopard, B. A lattice boltzmann study of blood flow in stented aneurism. *Future Generation Computer Systems*, 20(6):925–934, 2004.

[56] Hochmuth, R. and Waugh, R. Erythrocyte membrane elasticity and viscosity. *Annual review of physiology*, 49(1):209–219, 1987.

[57] Hou, S., Sterling, J., Chen, S., and Doolen, G. A lattice boltzmann subgrid model for high reynolds number flows. *arXiv preprint comp-gas/9401004*, 1994.

[58] Hoyas, S. and Jiménez, J. Scaling of the velocity fluctuations in turbulent channels up to reτ= 2003. *Physics of Fluids (1994-present)*, 18(1):011702, 2006.

[59] Hoyert, D. L., Xu, J., et al. Deaths: preliminary data for 2011. *National vital statistics reports*, 61(6):1–51, 2012.

[60] Humphrey, J. Vascular adaptation and mechanical homeostasis at tissue, cellular, and sub-cellular levels. *Cell biochemistry and biophysics*, 50(2):53–78, 2008.

[61] Humphrey, J. D., Milewicz, D. M., Tellides, G., and Schwartz, M. A. Dysfunctional mechanosensing in aneurysms. *Science*, 344(6183):477–479, 2014.

[62] Iaccarino, G., Ooi, A., Durbin, P., and Behnia, M. Reynolds averaged simulation of unsteady separated flow. *International Journal of Heat and Fluid Flow*, 24(2): 147–156, 2003.

[63] Issa, R. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 1986.

[64] Januszewski, M. and Kostur, M. Sailfish: A flexible multi-gpu implementation of the lattice boltzmann method. *Computer Physics Communications*, 185(9):2350 – 2368, 2014. ISSN 0010-4655. doi: http://dx.doi.org/10.1016/j.cpc.2014.04.018. URL `http://www.sciencedirect.com/science/article/pii/S0010465514001520`.

[65] Jeong, J. and Hussain, F. On the identification of a vortex. *Journal of fluid mechanics*, 285:69–94, 1995.

[66] Jimenez, J. and Moin, P. The minimal flow unit in near-wall turbulence. *Journal of Fluid Mechanics*, 225(213-240), 1991.

[67] Kang, X. and Dun, Z. Accuracy and grid convergence of wall shear stress measured by lattice boltzmann method. *International Journal of Modern Physics C*, 25(11), 2014.

[68] Karlin, I., Ferrante, A., and Öttinger, H. Perfect entropy functions of the lattice boltzmann method. *EPL (Europhysics Letters)*, 47(2):182, 1999.

[69] Karlin, I. V. and Succi, S. Equilibria for discrete kinetic equations. *Physical Review E*, 58:4053, 1998.

[70] Keating, B., Vahala, G., Yepez, J., Soe, M., and Vahala, L. Entropic lattice boltzmann representations required to recover navier-stokes flows. *Phys. Rev. E*, 75:036712, Mar 2007. doi: 10.1103/PhysRevE.75.036712. URL `http://link.aps.org/doi/10.1103/PhysRevE.75.036712`.

[71] Kerber, C. W. and Heilman, C. B. Flow dynamics in the human carotid artery: I. preliminary observations using a transparent elastic model. *American journal of neuroradiology*, 13(1):173–180, 1992.

[72] Kida, S. Three-dimensional periodic flows with high-symmetry. *Journal of the Physical Society of Japan*, 54(6):2132–2136, 1985. doi: 10.1143/JPSJ.54.2132. URL `http://jpsj.ipap.jp/link?JPSJ/54/2132/`.

[73] Kim, J., Moin, P., and Moser, R. Turbulence statistics in fully developed channel flow at low reynolds number. *Journal of Fluid Mechanics*, 177(1):133–166, 1987.

[74] Klewicki, J., Metzger, M., Kelner, E., and Thurlow, E. Viscous sublayer flow visualizations at r$\theta \approx$ 1 500 000. *Physics of Fluids (1994-present)*, 7(4):857–863, 1995.

[75] Kline, S., Reynolds, W., Schraub, F., and Runstadler, P. The structure of turbulent boundary layers. *Journal of Fluid Mechanics*, 30(04):741–773, 1967.

[76] Klöckner, A., Pinto, N., Lee, Y., Catanzaro, B., Ivanov, P., and Fasih, A. Pycuda and pyopencl: A scripting-based approach to gpu run-time code generation. *Parallel Computing*, 38(3):157–174, 2012.

[77] Kolmogorov, A. N. Dissipation of energy in the locally isotropic turbulence. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 434(1890):15–17, 1991. ISSN 0962-8444. doi: 10.1098/rspa.1991.0076.

[78] Kondo, S., Hashimoto, N., Kikuchi, H., Hazama, F., Nagata, I., and Kataoka, H. Cerebral aneurysms arising at nonbranching sites an experimental study. *Stroke*, 28(2):398–404, 1997.

[79] Krafczyk, M., Cerrolaza, M., Schulz, M., and Rank, E. . Analysis of 3d transient blood flow passing through an artificial aortic valve by lattice–boltzmann methods. *Journal of Biomechanics*, 31(5):453–462, 1998.

[80] Kreutzer, M. T., Kapteijn, F., Moulijn, J. A., and Heiszwolf, J. J. Multiphase monolith reactors: chemical reaction engineering of segmented flow in microchannels. *Chemical Engineering Science*, 60(22):5895–5916, 2005.

[81] Kreutzer, M. T., Kapteijn, F., Moulijn, J. A., Kleijn, C. R., and Heiszwolf, J. J. Inertial and interfacial effects on pressure drop of taylor flow in capillaries. *AIChE Journal*, 51(9):2428–2440, 2005.

[82] Krüger, T., Varnik, F., and Raabe, D. Shear stress in lattice boltzmann simulations. *Physical Review E*, 79(4):046704, 2009.

[83] Ku, D. N. Blood flow in arteries. *Annual Review of Fluid Mechanics*, 29(1): 399–434, 1997.

[84] Kupershtokh, A., Medvedev, D., and Karpov, D. On equations of state in a lattice boltzmann method. *Computers & Mathematics with Applications*, 58(5): 965 – 974, 2009. ISSN 0898-1221. doi: 10.1016/j.camwa.2009.02.024. URL http://www.sciencedirect.com/science/article/pii/S0898122109001011.

[85] Kurowski, K., Kulczewski, M., and Dobski, M. Parallel and gpu based strategies for selected cfd and climate modeling models. In *Information Technologies in Environmental Engineering*, pages 735–747. Springer, 2011.

[86] Kusumaatmaja, H. and Yeomans, J. M. Contact line dynamics in binary lattice boltzmann simulations. *Physical Review E*, 78(5):056709, 2008. URL http://pre.aps.org/abstract/PRE/v78/i5/e056709.

[87] Kuznik, F., Obrecht, C., Rusaouen, G., and Roux, J.-J. Lbm based flow simulation using gpu computing processor. *Computers & Mathematics with Applications*, 59 (7):2380 – 2392, 2010. ISSN 0898-1221. doi: 10.1016/j.camwa.2009.08.052. URL http://www.sciencedirect.com/science/article/pii/S0898122109006361.

[88] Ladd, A. J. C. Numerical Simulations of Particulate Suspensions via a Discretized Boltzmann Equation Part I. Theoretical Foundation. *Journal of Fluid Mechanics*, 271:285–309, July 1993. ISSN 0022-1120. doi: 10.1017/s0022112094001771. URL http://dx.doi.org/10.1017/s0022112094001771.

[89] Lammers, P., Beronov, K., Volkert, R., Brenner, G., and Durst, F. Lattice bgk direct numerical simulation of fully developed turbulence in incompressible plane channel flow. *Computers & fluids*, 35(10):1137–1153, 2006.

[90] Landau, L. and Lifshitz, E. Fluid mechanics, vol. 6. *Course of Theoretical Physics*, pages 227–229, 1987.

[91] Latt, J. and Chopard, B. Lattice boltzmann method with regularized pre-collision distribution functions. *Mathematics and Computers in Simulation*, 72(2):165–168, 2006.

[92] Liu, D. and Wang, S. Hydrodynamics of taylor flow in noncircular capillaries. *Chemical Engineering and Processing: Process Intensification*, 47(12):2098–2106, 2008.

[93] Malek, A. M., Alper, S. L., and Izumo, S. Hemodynamic shear stress and its role in atherosclerosis. *Jama*, 282(21):2035–2042, 1999.

[94] Markov, I. L. Limits on Fundamental Limits to Computation. *Nature*, 512 (7513):147–154, Aug. 2014. ISSN 0028-0836. doi: 10.1038/nature13570. URL http://dx.doi.org/10.1038/nature13570.

[95] Martinuzzi, R. and Tropea, C. The flow around surface-mounted, prismatic obstacles placed in a fully developed channel flow (data bank contribution). *Journal of Fluids Engineering*, 115(1):85–92, 1993.

[96] Marusic, I., McKeon, B., Monkewitz, P., Nagib, H., Smits, A., and Sreenivasan, K. Wall-bounded turbulent flows at high reynolds numbers: recent advances and key issues. *Physics of Fluids (1994-present)*, 22(6):065103, 2010.

[97] Matyka, M. and Koza, Z. How to calculate tortuosity easily? *arXiv preprint arXiv:1203.5646*, 2012.

[98] Matyka, M., Koza, Z., Gołembiewski, J., Kostur, M., and Januszewski, M. Anisotropy of flow in stochastically generated porous media. *Physical Review E*, 88(2):023018, 2013.

[99] Matyka, M., Koza, Z., and Mirosław, Ł. Wall orientation and shear stress in the lattice boltzmann model. *Computers & Fluids*, 73:115–123, 2013.

[100] Mei, R., Luo, L.-S., Lallemand, P., and d'Humières, D. Consistent initial conditions for lattice Botlzmann simulations. *Comput. Fluids*, 35:855–862, 2006.

[101] Meinders, E. and Hanjalić, K. Vortex structure and heat transfer in turbulent flow over a wall-mounted matrix of cubes. *International Journal of Heat and Fluid Flow*, 20(3):255–267, 1999.

[102] Meinders, E. and Hanjalić, K. Experimental study of the convective heat transfer from in-line and staggered configurations of two wall-mounted cubes. *International Journal of Heat and mass transfer*, 45(3):465–482, 2002.

[103] Meinders, E., Hanjalic, K., and Martinuzzi, R. Experimental study of the local convection heat transfer from a wall-mounted cube in turbulent channel flow. *Journal of heat transfer*, 121(3):564–573, 1999.

[104] Melchionna, S., Amati, G., Bernaschi, M., Bisson, M., Succi, S., Mitsouras, D., and Rybicki, F. J. Risk assessment of atherosclerotic plaques based on global biomechanics. *Medical engineering & physics*, 35(9):1290–1297, 2013.

[105] Moreno, R. and Smedby, O. Volume-based fabric tensors through lattice-boltzmann simulations. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 3179–3184. IEEE, 2014.

[106] Moser, R. D., Kim, J., and Mansour, N. N. Direct numerical simulation of turbulent channel flow up to re= 590. *Physics of fluids*, 11:943, 1999.

[107] Oberman, S. F. and Siu, M. Y. A high-performance area-efficient multifunction interpolator. In *Computer Arithmetic, 2005. ARITH-17 2005. 17th IEEE Symposium on*, pages 272–279. IEEE, 2005.

[108] Obrecht, C., Kuznik, F., Tourancheau, B., and Roux, J.-J. A new approach to the lattice boltzmann method for graphics processing units. *Computers & Mathematics with Applications*, 61(12):3628 – 3638, 2011. ISSN 0898-1221. doi: 10.1016/j.camwa.2010.01.054. URL `http://www.sciencedirect.com/science/article/pii/S089812211000091X`.

[109] Obrecht, C., Kuznik, F., Tourancheau, B., and Roux, J.-J. Multi-gpu implementation of the lattice boltzmann method. *Computers & Mathematics with Applications*, (0):–, 2011. ISSN 0898-1221. doi: 10.1016/j.camwa.2011.02.020. URL `http://www.sciencedirect.com/science/article/pii/S0898122111001064`.

[110] Obrecht, C., Kuznik, F., Tourancheau, B., and Roux, J.-J. Global memory access modelling for efficient implementation of the lattice boltzmann method on graphics processing units. In *Proceedings of the 9th international conference on High performance computing for computational science*, VECPAR'10, pages 151–161, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-19327-9. URL `http://dl.acm.org/citation.cfm?id=1964238.1964257`.

[111] OpenCFD. OpenFOAM, http://www.openfoam.com/. Web Site, 2012. URL `http://www.openfoam.com/`.

[112] Palabos. URL `http://www.palabos.org`.

[113] Pan, W., Fedosov, D. A., Caswell, B., and Karniadakis, G. E. Predicting dynamics and rheology of blood flow: a comparative study of multiscale and low-dimensional models of red blood cells. *Microvascular research*, 82(2):163–170, 2011.

[114] Patankar, S. and Spalding, D. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787 – 1806, 1972.

[115] Pereira, G. G., Dupuy, P. M., Cleary, P. W., and Delaney, G. W. Comparison of permeability of model porous media between sph and lb. *Progress in Computational Fluid Dynamics, an International Journal*, 12(2):176–186, 2012.

[116] Peters, A., Melchionna, S., Kaxiras, E., Lätt, J., Sircar, J., Bernaschi, M., Bison, M., and Succi, S. Multiscale simulation of cardiovascular flows on the ibm bluegene/p: Full heart-circulation system at red-blood cell resolution. In *Proceedings of the 2010 ACM/IEEE international conference for high performance computing, networking, storage and analysis*, pages 1–10. IEEE Computer Society, 2010.

[117] Pooley, C. M. and Furtado, K. Eliminating spurious velocities in the free-energy lattice boltzmann method. *Phys. Rev. E*, 77:046702, Apr 2008. doi: 10.1103/PhysRevE.77.046702. URL `http://link.aps.org/doi/10.1103/PhysRevE.77.046702`.

[118] Pooley, C. M., Kusumaatmaja, H., and Yeomans, J. M. Contact line dynamics in binary lattice boltzmann simulations. *Phys. Rev. E*, 78:056709, Nov 2008. doi: 10.1103/PhysRevE.78.056709. URL `http://link.aps.org/doi/10.1103/PhysRevE.78.056709`.

[119] Reneman, R. S., Arts, T., and Hoeks, A. Wall shear stress–an important determinant of endothelial cell function and structure–in the arterial system in vivo. discrepancies with theory. *Journal of vascular research*, 43(3):251–269, 2005.

[120] Rodi, W. Comparison of les and rans calculations of the flow around bluff bodies. *Journal of Wind Engineering and Industrial Aerodynamics*, 69:55–75, 1997.

[121] Rodi, W., Ferziger, J., Breuer, M., and Pourquie, M. Status of large eddy simulation: results of a workshop. *Transactions-American Society of Mechanical Engineers Journal of Fluids Engineering*, 119:248–262, 1997.

[122] Rupp, K., Rudolf, F., and Weinbub, J. ViennaCL - A High Level Linear Algebra Library for GPUs and Multi-Core CPUs. In *Intl. Workshop on GPUs and Scientific Applications*, pages 51–56, 2010.

[123] Samijo, S., Willigers, J., Barkhuysen, R., Kitslaar, P., Reneman, R., Brands, P., and Hoeks, A. Wall shear stress in the human common carotid artery as function of age and gender. *Cardiovascular Research*, 39(2):515–522, 1998. ISSN 0008-6363. doi: 10.1016/S0008-6363(98)00074-1.

[124] Schubert, T., Santini, F., Stalder, A., Bock, J., Meckel, S., Bonati, L., Markl, M., and Wetzel, S. Dampening of blood-flow pulsatility along the carotid siphon: does form follow function? *American Journal of Neuroradiology*, 32(6):1107–1112, 2011.

[125] Schünke, M., Ross, L., Schulte, E., Schumacher, U., and Lamperti, E. *Thieme Atlas of Anatomy: General Anatomy and Musculoskeletal System*. Thieme Atlas of Anatomy. Thieme, 2006. ISBN 9783131420817. URL `https://books.google.co.jp/books?id=NK9TgTaGt6UC`.

[126] Shan, X. and Chen, H. Lattice boltzmann model for simulating flows with multiple phases and components. *Physical Review E*, 47 (3):1815–1819, Mar 1993. doi: 10.1103/PhysRevE.47.1815. URL `http://dx.doi.org/10.1103/PhysRevE.47.1815`.

[127] Shan, X. W. and He, X. Y. Discretization of the velocity space in the solution of the boltzmann equation. *Physical Review Letters*, 80(1):65–68, 1998.

[128] Skordos, P. A. Initial and boundary conditions for the lattice boltzmann method. *Phys. Rev. E*, 48:4823–4842, Dec 1993. doi: 10.1103/PhysRevE.48.4823. URL `http://link.aps.org/doi/10.1103/PhysRevE.48.4823`.

[129] Smagorinsky, J. General circulation experiments with the primitive equations: I. the basic experiment*. *Monthly weather review*, 91(3):99–164, 1963.

[130] Smith, C. and Metzler, S. The characteristics of low-speed streaks in the near-wall region of a turbulent boundary layer. *Journal of Fluid Mechanics*, 129:27–54, 1983.

[131] Spasov, M., Rempfer, D., and Mokhasi, P. Simulation of a turbulent channel flow with an entropic lattice boltzmann method. *International journal for numerical methods in fluids*, 60(11):1241–1258, 2009.

[132] Sreenivasan, K. The turbulent boundary layer. In *Frontiers in experimental fluid mechanics*, pages 159–209. Springer, 1989.

[133] Stahl, B., Chopard, B., and Latt, J. Measurements of wall shear stress with the lattice boltzmann method and staircase approximation of boundaries. *Computers & Fluids*, 39(9):1625–1633, 2010.

[134] Swift, M., Orlandini, E., Osborn, W., and Yeomans, J. Lattice Boltzmann simulations of liquid-gas and binary fluid systems. *Phys. Rev. E*, 54(5):5041–5052, 1996.

[135] Taylor, C. A. and Draney, M. T. Experimental and computational methods in cardiovascular fluid mechanics. *Annu. Rev. Fluid Mech.*, 36:197–231, 2004.

[136] Taylor, C. A. and Figueroa, C. Patient-specific modeling of cardiovascular mechanics. *Annual review of biomedical engineering*, 11:109–134, 2009.

[137] Thulasidas, T., Abraham, M., and Cerro, R. Bubble-train flow in capillaries of circular and square cross section. *Chemical Engineering Science*, 50(2):183–199, 1995.

[138] Tikhomirov, V. On the degeneration of isotropic turbulence in an incompressible viscous fluid. In Tikhomirov, V., editor, *Selected Works of A. N. Kolmogorov*, volume 25 of *Mathematics and Its Applications (Soviet Series)*, pages 319–323. Springer Netherlands, 1991. ISBN 978-94-010-5347-1. doi: 10.1007/978-94-011-3030-1_46. URL `http://dx.doi.org/10.1007/978-94-011-3030-1_46`.

[139] Tolke, J. Implementation of a Lattice Boltzmann kernel using the Compute Unified Device Architecture developed by nVIDIA. *Comput. Visual Sci.*, 13(1):29–39, 2008.

[140] Tölke, J. and Krafczyk, M. TeraFLOP computing on a desktop PC with GPUs for 3D CFD. *Int. J. Comput. Fluid Dyn.*, 22(7):443–456, 2008.

[141] Tomczak, T. Acceleration of iterative navier-stokes solvers on graphics processing units. *International Journal Computational Fluid Dynamics*, 2012.

[142] Tosi, F., Ubertini, S., Succi, S., and Karlin, I. V. Optimization strategies for the entropic lattice boltzmann method. *Journal of Scientific Computing*, 30(3): 369–387, 2007.

[143] Valencia, A. A., Guzman, A. M., Finol, E. A., and Amon, C. H. Blood flow dynamics in saccular aneurysm models of the basilar artery. *Journal of Biomechanical Engineering*, 2006.

[144] van Prehn, J., Vincken, K., Sprinkhuizen, S., Viergever, M., van Keulen, J., van Herwaarden, J., Moll, F., and Bartels, L. Aortic pulsatile distention in young healthy volunteers is asymmetric: Analysis with ecg-gated {MRI}. *European Journal of Vascular and Endovascular Surgery*, 37(2):168 – 174, 2009. ISSN 1078-5884. doi: http://dx.doi.org/10.1016/j.ejvs.2008.11.007. URL `http://www.sciencedirect.com/science/article/pii/S1078588408005935`.

[145] Verberg, R. and Ladd, A. Simulation of low-reynolds-number flow via a time-independent lattice-boltzmann method. *Physical Review E*, 60(3):3366, 1999.

[146] Vorp, D. A., Steinman, D. A., and Ethier, C. R. Computational modeling of arterial biomechanics. *Computing in Science & Engineering*, 3(5):51–64, 2001.

[147] Vratis. Arael / SpeedIT Flow, 2012. URL `http://www.vratis.com/`.

[148] Vreman, A. and Kuerten, J. Comparison of direct numerical simulation databases of turbulent channel flow at reτ= 180. *Physics of Fluids (1994-present)*, 26(1): 015102, 2014.

[149] Wallace, J. M., Eckelmann, H., and Brodkey, R. S. The wall region in turbulent shear flow. *Journal of Fluid Mechanics*, 54(01):39–48, 1972.

[150] Wei, Z., Yong, W., and Yue-Hong, Q. Stability analysis for flow past a cylinder via lattice boltzmann method and dynamic mode decomposition. 2015.

[151] White, F. M. and Corfield, I. *Viscous fluid flow*, volume 3. McGraw-Hill New York, 2006.

[152] Womersley, J. R. Method for the calculation of velocity, rate of flow and viscous drag in arteries when the pressure gradient is known. *The Journal of physiology*, 127(3):553–563, 1955.

[153] Wong, H., Radke, C., and Morris, S. The motion of long bubbles in polygonal capillaries. part 1. thin films. *Journal of Fluid Mechanics*, 292:71–94, 1995.

[154] Wong, H., Radke, C., and Morris, S. The motion of long bubbles in polygonal capillaries. part 2. drag, fluid pressure and fluid flow. *Journal of Fluid Mechanics*, 292:95–110, 1995.

[155] Xiu-Ying, K., Yu-Pin, J., Da-He, L., and Yong-Juan, J. Three-dimensional lattice boltzmann method for simulating blood flow in aortic arch. *Chinese Physics B*, 17(3):1041, 2008. URL `http://stacks.iop.org/1674-1056/17/i=3/a=049`.

[156] Yakhot, A., Anor, T., Liu, H., and Nikitin, N. Direct numerical simulation of turbulent flow around a wall-mounted cube: spatio-temporal evolution of large-scale vortices. *Journal of Fluid Mechanics*, 566:1–9, 2006.

[157] Yakhot, A., Liu, H., and Nikitin, N. Turbulent flow around a wall-mounted cube: A direct numerical simulation. *International journal of heat and fluid flow*, 27(6): 994–1009, 2006.

[158] Yang, Z., Palm, B., and Sehgal, B. Numerical simulation of bubbly two-phase flow in a narrow channel. *International Journal of Heat and Mass Transfer*, 45(3): 631–639, 2002.

[159] Yue, J., Luo, L., Gonthier, Y., Chen, G., and Yuan, Q. An experimental study of air–water taylor flow and mass transfer inside square microchannels. *Chemical Engineering Science*, 64(16):3697–3708, 2009.

[160] Závodszky, G. and Paál, G. Validation of a lattice boltzmann method implementation for a 3d transient fluid flow in an intracranial aneurysm geometry. *International Journal of Heat and Fluid Flow*, 44:276–283, 2013.

[161] Zou, Q. and He, X. On pressure and velocity boundary conditions for the lattice boltzmann bgk model. *Physics of Fluids*, 9:1591, 1997.